NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

IMPROVING OPERATIONAL EFFECTIVENESS OF TACTICAL LONG ENDURANCE UNMANNED AERIAL SYSTEMS (TALEUAS) BY UTILIZING SOLAR POWER

by

Nahum Camacho

June 2014

Thesis Advisor:	Vladimir N. Dobrokhodov
Co-Advisor:	Kevin D. Jones
Second Reader:	Isaac Kaminer

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** June 2014	**3. REPORT TYPE AND DATES COVERED** Engineer's Thesis	
4. TITLE AND SUBTITLE IMPROVING OPERATIONAL EFFECTIVENESS OF TACTICAL LONG ENDURANCE UNMANNED AERIAL SYSTEMS (TALEUAS) BY UTILIZING SOLAR POWER			**5. FUNDING NUMBERS**
6. AUTHOR(S) Nahum Camacho			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) E2O-MARINE CORPS CRUSER ARL ONR			**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol number ____N/A____.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited			**12b. DISTRIBUTION CODE**

13. ABSTRACT (maximum 200 words)

This thesis develops, implements, and validates a hybrid energy-harvesting technique that enables extracting energy from the environment by utilizing convective thermals as a source of potential energy, and exploiting solar radiation for photovoltaic (PV) energy to achieve long endurance flight of an autonomous glider. The dynamic behavior of convective thermals, as well as their mathematical models, are studied to determine their motion, while the navigation task is simultaneously solved using a Bayesian search approach that is based on the prior knowledge of the 3D elevation. This study advances an existing technique for detection of thermals by implementing the online identification of the airplane sink rate polar. The glider's climb rate is optimized by implementing a modified thermalling controller, and its performance is compared to an existing method of centering in thermals. The integration of the energy extracted from the solar radiation is accomplished by the design of an Electrical Energy Management System (EEMS) that safely collects and distributes the energy onboard. The electrical energy is supplied by the semi-rigid mono crystalline silicon solar cells, which are embedded into the skin of the glider's wings without distorting the airfoil.

To validate and verify the algorithms developed in MATLAB/Simulink, an interface to a high-fidelity pilot's training flight simulator was designed. Furthermore, the numerical algorithms were integrated onboard a prototype SB-XC glider equipped with solar cells to enable the desired energy-harvesting technique. Flight test results verify the feasibility of the developed algorithms.

14. SUBJECT TERMS: convective thermals, thermalling control, system identification, photovoltaics, Bayesian search, guidance, navigation, path planning, Electrical Energy Management System, MATLAB/Simulink, mathematical modeling and simulation.			**15. NUMBER OF PAGES** 185
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

IMPROVING OPERATIONAL EFFECTIVENESS OF TACTICAL LONG ENDURANCE UNMANNED AERIAL SYSTEMS (TALEUAS) BY UTILIZING SOLAR POWER

Nahum Camacho
Lieutenant, Mexican Navy
B.S., School of Engineering of the Mexican Navy, 2007

Submitted in partial fulfillment of the
requirements for the degree of

MECHANICAL ENGINEER

and

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
June 2014

Author:	Nahum Camacho
Approved by:	Vladimir N. Dobrokhodov Thesis Advisor
	Kevin D. Jones Co-Advisor
	Isaac Kaminer Second Reader
	Knox T. Millsaps Chair, Department of Mechanical & Aerospace Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This thesis develops, implements, and validates a hybrid energy-harvesting technique that enables extracting energy from the environment by utilizing convective thermals as a source of potential energy, and exploiting solar radiation for photovoltaic (PV) energy to achieve long endurance flight of an autonomous glider. The dynamic behavior of convective thermals, as well as their mathematical models, are studied to determine their motion, while the navigation task is simultaneously solved using a Bayesian search approach that is based on the prior knowledge of the 3D elevation. This study advances an existing technique for detection of thermals by implementing the online identification of the airplane sink rate polar. The glider's climb rate is optimized by implementing a modified thermalling controller, and its performance is compared to an existing method of centering in thermals. The integration of the energy extracted from the solar radiation is accomplished by the design of an Electrical Energy Management System (EEMS) that safely collects and distributes the energy onboard. The electrical energy is supplied by the semi-rigid mono crystalline silicon solar cells, which are embedded into the skin of the glider's wings without distorting the airfoil.

To validate and verify the algorithms developed in MATLAB/Simulink, an interface to a high-fidelity pilot's training flight simulator was designed. Furthermore, the numerical algorithms were integrated onboard a prototype SB-XC glider equipped with solar cells to enable the desired energy-harvesting technique. Flight test results verify the feasibility of the developed algorithms.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

API	application programming interface
ARPA-E	Advanced Research Projects Agency – Energy
CPU	central processing unit
DAQ	data acquisition
DCM	direction cosine matrix
EEMS	electrical energy management system
EKF	extended Kalman filter
EMS	energy management system
FOM	figure of merit
GNC	guidance, navigation and control
IFAC	International Federation of Automatic Control
IFC	interim flight clearance
IO	input/output
IR	infrared
IROS	International Conference on Intelligent Robots and Systems
ISR	Intelligence, Surveillance, and Reconnaissance
JIFX	Joint Interagency Field Exploration
KF	Kalman filter
MPPT	Maximum Power Point Tracker
MSL	mean sea level
NLF	natural laminar flow
NREL	National Renewable Energy Lab
PCM	protection circuit module
POI	points of interest
PV	photovoltaic
RLLS	recursive linear least squares
SIL	software in the loop
SOC	state of charge
TaLEUAS	tactical long-endurance unmanned aerial system
TAS	True Air Speed

TEK	Total Energy Compensated
TSP	traveling salesman problem
UAV	unmanned aerial vehicle
UDP	user datagram protocol
UKF	unscented Kalman filter
6DoF	six degrees of freedom

ACKNOWLEDGMENTS

I first and foremost thank God for providing protection and being my strength and guide at every moment of my life.

My deepest appreciation goes to my wife, Maria Elena. Her unconditional love, support, encouragement, and patience have always been present in any challenge that I face.

I am in debt to the Mexican Navy for giving me such a great opportunity of being at NPS to improve my professional and military careers; mainly, I would like to thank CDR Mariano Lizarraga and CDR Miguel Alvarado, both NPS graduates who believed in me for representing my country.

I want to thank Drs. Vladimir Dobrokhodov, Kevin Jones, and Isaac Kaminer; they are the builders of my knowledge. Their wisdom and cutting-edge ideas make utopian projects come to life. I really appreciate their advice and the countless hours dedicated to me and this project.

I would also like to express my gratitude to the Consortium of Robotics and Unmanned System Educational Research and the Marine Corps for their financial support. Their contribution made possible the completion of this project.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

Flight endurance is one of the most important characteristics of an airplane when planning the tactical missions that an Unmanned Air Vehicle (UAV) is able to carry out.

This thesis develops and implements an energy-harvesting technique using solar radiation in two ways: convective thermals, i.e., soaring, and photovoltaic energy. The combination of these two phenomena will enable 24/7 flying capabilities.

Technological advances in electronics are merged with state-of-the-art control algorithms to achieve the integration of a fully autonomous glider capable of using the previously mentioned solar radiation, and apply it in a multiday mission.

Some articles and papers have shown the feasibility of implementing separately the use of the two phenomena produced by solar radiation; however, the combination of both has the potential to achieve 24/7 flights. Validation and verification of this combination is provided by the software in the loop simulations (SIL) and real flight tests conducted in this study.

A. ENVISIONED MISSION OF MULTIPLE AUTONOMOUSLY SOARING GLIDERS

Imagine a conflict taking place behind enemy lines. To provide reconnaissance over the area, a squad of autonomous cooperative gliders is deployed from a remote location. Each glider carries a specific sensor/weapon to give air support to friendly forces. At the same time, a communication network is established automatically by the squad, and encrypted video/data of the conflict is sent to the command center miles away in a safe area. The gliders will remain airborne day and night in multiday missions as required, without any power supply or human close supervision.

The energy required to support the long endurance mission is provided by the sun to the aircraft in two ways: first by autonomous soaring in the columns of convective air that are usually called “thermals”, and second by storing electrical energy produced by the photovoltaic cells covering the wings in the onboard batteries.

The flock of gliders operates as a neural network wherein each element communicates with the neighboring gliders. Information gained by the gliders during their mission is shared among them and also concentrated in a remote, centralized, intelligent ground-control station.

The squad can be thought of as a set of low-altitude satellites with the potential for performing a number of useful missions; transition to a different mission can be done either by installing different sensors and payloads onboard or by employing a set of heterogeneous gliders carrying various sensors/payloads.

After the squad is assigned a mission, all its component gliders distribute their sensing and payload capabilities automatically; this is done to accomplish that mission over an extended period of time, and to harvest the maximum solar energy available to enable that extended operational endurance. The autonomous gliders might be assigned a wide variety of missions:

- persistent Intelligence, Surveillance, and Reconnaissance (ISR)
- encrypted network with video and communication coverage
- high-resolution terrain mapping
- target search with geolocation and designation capabilities
- target tracking
- convoy protection
- jamming and deception
- early warning of approaching enemy to protect friendly ground forces

B. THESIS OBJECTIVES

The key objective of this thesis is to develop an extended endurance capability of a single autonomous glider by enabling its onboard systems to harvest thermal and solar energy.

As a logical extension, the thesis explores the technological solutions of integrating multiple autonomous thermal-soaring solar-powered gliders in a cooperative mission to:

- significantly increase their capability of rapidly finding the convective thermals
- complement their operational capabilities in persistent ISR missions

C. FORMULATION OF THE PROBLEM

In order to accomplish these objectives it is necessary to integrate in a single platform multiple complementary capabilities:

- thermal searching
- thermal detection
- autonomous soaring
- photovoltaic energy harvesting

When these capabilities are developed and integrated onboard a single platform, the automatic formation of a team of fully autonomous gliders capable of staying airborne in a multiday flight becomes a reality.

D. PREVIOUS STUDIES AND APPLIED RESEARCH

Achieving success in this project requires systematic development and integration of knowledge from several adjacent areas of expertise, with all of them contributing to the same goal; that is, a design of an aerodynamically efficient airplane that is capable of autonomously harvesting energy from the sun. The expected contributions from the various engineering disciplines are as follows:

- Airframe optimization. Any reduction in drag increases endurance on an airplane; i.e., with the same amount of energy the UAV will travel a larger distance or will stay airborne longer. Another characteristic desired in such an optimized platform is a high lift with low induced drag at the cruise speed.
- High efficiency photovoltaic cells and high-specific energy batteries. One of the energy sources to be exploited by the aircraft is the solar radiation that is collected onboard through the photovoltaic energy extraction. During the day, this energy is used to power the avionics onboard, while excess energy is stored in a set of onboard batteries.

During the night, the stored energy powers avionics onboard and is used for propulsion as required.

- Convective thermal search and detection. Autonomous soaring relies on the existence of convective air that first needs to be sought and detected. Utilizing convective thermals allows significant gain of potential energy through altitude while not using any energy for propulsion. Utilizing the typically present INS/GPS sensing capabilities onboard as well as the algorithms of energy estimation, it is possible to find thermals and determine their potential utility.
- Identification of convective thermals. Knowledge of the convective air characteristics determines a number of utility parameters; among them are the intensity of the updraft, the maximum achievable ceiling, the geographic location and its motion. All these characteristics are essential when planning and executing a single glider mission. They are even more important in planning and executing the multiple cooperative glider missions.
- Autonomous soaring. This capability allows the glider to gain altitude without spending energy for propulsion. In thermal soaring approach, the columns of rising air (i.e., thermal updrafts) are used as energy sources.
- Cooperative flight. Since the columns of rising air are invisible, finding their location is crucial for the extended flight operation. Assuming that every airplane has the same mobility and sensing characteristics, the greater the number of gliders, the quicker the search and more reliable detection, and identification of updrafts will be.

1. Airframe Optimization

Airframe optimization is typically done by minimizing the parasitic drag induced by the air passing the airframe, and maximizing the lift produced primarily by the wings and fuselage. Throughout the years, research has been done on these two components in order to produce high-efficiency airframes.

Back in 1968, Galvao [1] studied a methodology to get low drag three-dimensional bodies; it was based on the principle that a majority of glider fuselages have plan and side forms similar to low drag airfoils. Galvao applied potential flow and boundary layer theories for developing geometries that favor laminar flow downstream by creating prescribed negative pressure gradients. In theory, glider fuselages obey these design criteria.

Further studies carried in 1984 by Dodbele and Van Dam [2] had as their main objective the design of bodies with a large region of Natural Laminar Flow (NLF). Feasibility of this objective was motivated by technological advances in material sciences which allowed for creation of surfaces with minimal roughness. As a result, a computational design procedure was developed to obtain a low-drag body shape. However, the comparison of computed and experimental boundary layer transition points showed a mismatch in the results. One of the reasons for this mismatch was attributed to the inappropriately measured experimental data; and as solution, it was suggested that a full-scale high-Reynolds number wind tunnel experiment be conducted to obtain qualitatively and quantitatively better transition data. Nowadays, new technological advances in material sciences together with novel manufacturing process produce lighter and more resistant airframes with large regions of NLF as in Tactical Long Endurance Unmanned Aerial Systems (TaLEUAS).

Research in the optimization of airfoils has been done by Boermans and Garrel [3], where the airfoils were designed, and the effect of wings made with them was analyzed for multiple plan-form fuselages. The results achieved several goals, such as a low drag for prescribed ranges of lift coefficients and Reynolds numbers, improved climbing characteristics, graceful and predictable stall characteristics, laminar flow extended in some cases up to 95% of the chord length (Figure 1).

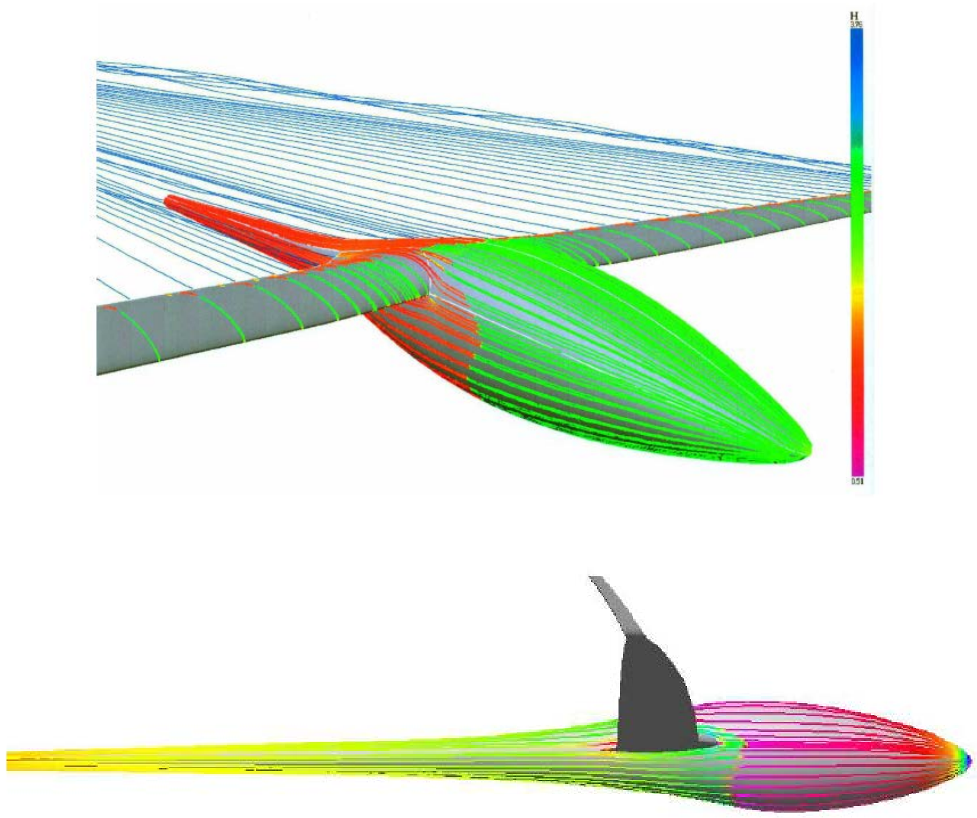


Figure 1. Laminar flow over fuselage of an optimized glider airframe, from [3].

Selig, Guglielmo, Broerer and Giguère published in 1995 the first edition of the Summary of Low Speed Airfoil Data [4], which later would become a series of volumes that document the experimental results of tests performed on the low Reynolds number airfoils at the University of Illinois at Urbana-Champaign. This kind of airfoils is used nowadays in several aircrafts including the gliders like TaLEUAS.

In 1997 Boermans and Nicolosi [5] investigated the effect of modifying fuselage geometry dimensions on the produced drag; the work utilized a three-dimensional panel code. This research demonstrated that increase of length of the cockpit (to allow for additional instrumentation) within a certain range has no impact on the resulting drag, while an increase of height of about 10% will increase the fuselage drag by about 13%.

A design methodology used to create high-lift low-Reynolds number airfoils was established back in 1997 by Selig and Guglielmo [6]. The development of these airfoils represented great potential advantages for military applications such as: carrying heavy payloads, shortening takeoff and landing distances, and lowering stall speed. The main idea in the airfoil design was the use of concave pressure recovery with aft loading. For example, the new airfoil S1223 for a Reynolds number of 2×10^5 was created and validated with wind tunnel tests getting a maximum lift coefficient of 2.2. This coefficient was higher than the one obtained for similar airfoils, such as the FX 63-137, with a maximum lift coefficient of 1.7 for the same Reynolds number.

Another design methodology based on inverse airfoil design was applied by Gopalarathnam and Selig [7] in 2001. The main goal this time was to ensure NLF over the airfoil. The developed multipoint inverse airfoil design method allows specifying the velocity and boundary layer properties over different portions of the airfoil. Further adjustment of the described parameters results in families of airfoils with desired different lift, drag, and pitching moment properties.

2. Autonomous Soaring

Thermal soaring is a technique that involves the use of columns of rising air to gain altitude without using propulsion; it has always been used by some bird species, and lately by human pilots. One of the first attempts to replicate this technique in UAVs was developed by Allen [8]; in 2005, he developed and ran a computer simulation of autonomous soaring based on real data. The goal of this simulation was to determine the improvement in flight endurance gained by a small UAV as a result of utilizing the thermal soaring technique. Allen took atmospheric data from the Desert Rock area of Nevada; it was used to simulate the characteristics of the thermals in a parametric mathematical model. Variable characteristics of thermals in this model were size, strength, spacing, shape, and maximum height. Allen’s results showed that UAV endurance could be improved by up to 12 hours of flight just by using thermal soaring. The major contributions of this work are the development and implementation of mathematical models of thermals in simulations and the quantitative prediction of the extended endurance.

In 2007 Allen and Lin [9] designed and implemented onboard UAV, and flight tested a guidance and control method for autonomous soaring. The algorithms were based on the concept of total energy, where potential and kinetic energies were combined. The total energy was used to estimate position, radius, drift, and strength of thermals; later, these parameters were used to guide the glider in a circular path to enable autonomous thermal soaring.

Edwards [10] designed and implemented onboard a glider UAV a thermal locating and guidance algorithm based on the rate of change of the vehicle energy and the glider speed polar. Numerical techniques like adaptive grids and nonlinear regressions and correlations were applied in this research to get a better estimation of the thermal location. Also the author implemented the optimization of commanded speed based on the MacCready Speed-to-Fly theory, which is widely used in competition of cross-country glider flights.

In 2010, Akthar [11] explored the potential effect of integrating of several sensors to detect thermals; the effort led to the design and onboard integration of a Total Energy-compensated variometer and an infrared (IR) camera. These sensors were combined with weather information, and guidance and navigation algorithms. The objectives of this work were in the localization of hot areas on the ground using the IR camera, and generation of real-time trajectories for dynamic soaring utilizing the Inverse Dynamics Virtual Domain technique.

Andersson et al. [12] in 2012 analyzed the stability properties of the thermal centering control algorithm that was based on the total energy concept [9] and its first two time derivatives. Such a controller captured Reichmann’s climbing technique [13], where bank angle is actively adjusted to enable the glider to automatically soar around the center of the thermal at an optimal turning radius. Flight tests proved the reliability and high utility characteristics of the controller when tracking stationary and moving thermals.

3. Cooperative Flight

In 2008, Hanson [14] performed a study to demonstrate that cooperative behaviors found in flocks of birds could be applied to flocks of UAVs. These cooperative behaviors were implemented in UAVs by adopting the concept of potential fields that represent the mission objectives and environmental obstacles as attractive and repulsive forces. Despite the simplicity of the approach, these forces were able to deliver good robust navigation solutions and produce suitable spacing among the UAVs to guarantee the desired collision avoidance. In the application to the soaring flight, when a thermal is detected and identified all the members of the flock are attracted to it. Even though the result was based on real flocks' behaviors applied to UAVs, it was not tested onboard real glider platforms.

The algorithms that enable cooperation of multiple UAVs in thermal search, detection, and collaborative exploitation were developed, simulated, and tested in real flight by Andersson et al. in their research [15] that was first published in 2009. The objective was to determine the benefits of cooperative search for thermals by using two gliders flying over a bounded area of operation. Final flight test results confirmed that the likelihood of finding thermals is significantly increased; hence the UAVs' extended autonomy by minimizing time spent to find at least one thermal.

Antal et al. [16] in 2010 published results of numerical study where he developed and simulated an algorithm capable of identifying the center of a thermal. This algorithm was based on the Simultaneous Perturbation Stochastic Approximation approach and used gradients of the vertical speed to determine the optimal estimating solution for the center of the thermal. This algorithm was verified in a numerical simulation of a group of UAVs flying simultaneously and providing synchronous measurements of the same physical event. The mathematical model of a thermal adapted in [16] is the one previously developed by Allen in [8], who was the first applying the parametric models of thermals in the tasks of autonomous soaring. The work of Antal [16] modified Allen's approach by accounting for a decaying strength of updrafts over time. Results of this work showed in simulation the performance of a group of UAVs cooperating to find the center of a thermal.

In 2013, Cobano et al. in their work [17] applied a Bounded Recursive Search Algorithm to extend the endurance of a glider by gaining altitude using thermals located in known positions. The flight path of the glider was continuously updated onboard; such flight path was based on a series of waypoints corresponding to points of interest and the location of known thermals. As a result, the glider visited the points of interest while it used the thermals that were closest to its path. This demonstrates a mission application of an UAV that can harness energy from thermals while at the same time visit points of interest.

4. Parameter Estimation of Single Thermal Updraft

All the studies related to autonomous soaring are based on formal mathematical representation of physical characteristics of thermals; most often the model was an algebraic function parameterized by a finite set of values which represented the most significant physical features of an updraft. In order to better represent a real thermal, the mathematical models must be complex enough to capture all those characteristics.

Several approaches in estimation of the parameters of thermals have been used, including the Kalman Filter and its numerous modifications. In 2010, Hazard [18] utilized the Unscented Kalman Filter (UKF) to determine position, size and strength of a simulated thermal. Even though this type of filter assumes the same Gaussian characteristics of noise distribution as other Kalman Filters (KF), it is more accurate and can manage highly complex nonlinear equations relating the measurements and the states of the system without the computationally demanding efforts of calculating the underlying Jacobian as in the extended KF (EKF) version. Even though the UKF might handle highly complex nonlinear mathematical models, Hazard's implementation was based on a simple model, which could be changed for a more complex model that better captures the behavior of thermals and still have an accurate estimation of its characteristics by using the UKF.

5. Combination of Energy Harvesting Methods

Anton in 2008 [19] proposed a combination of energy harvesting methods using two sources. The objective of this project was to demonstrate that energy could be harvested using several methods. The first source was vibrational energy collected with piezoelectric patches placed at the roots of the glider’s wings and a cantilevered piezoelectric beam allocated in the fuselage. In this way, vibrations coming from the wings and fuselage could be used to generate electricity for onboard consumption. The second source was solar radiation captured by thin-film photovoltaic (PV) cells. The result of this combination showed that reasonable amounts of energy can be gained and stored onboard separately. Anton’s purpose was not to achieve a long endurance flight, but to demonstrate that more than one alternative energy source can be harnessed at the same time.

In 2011, Barnes et al. [20] proposed an approach to harvest energy using three different methods. The first method assumed capturing energy from solar radiation. The second method required the use of autonomous thermalling algorithms to gain altitude, hence increasing the potential energy. The third method was the implementation of a technique called regenerative soaring, where the motor of the UAV was used to generate electrical energy when the airplane was thermalling. Barnes’ approach was a good proposal, but has not been implemented yet.

TaLEUAS will implement two powerful sources of energy that separately have been demonstrated to extend the endurance of the UAVs where they have been implemented. Future improvements to TaLEUAS might be the addition of devices and algorithms to include more energy sources such as piezoelectric and regenerative soaring.

6. Historical Review of Battery Technology

Batteries onboard provide the backup electrical energy for powering the avionics, payload, and propeller. Their specific energy—capacity/weight ratio—is one of the most important parameters for selecting a battery and has been increasing its value throughout the years since the creation of the early batteries. The chemistry used on the batteries determines the specific energy among other performance characteristics; therefore, an analysis of the creation and evolution of the battery technology (Table 1) as well as the improvements in the specific energy (Figure 2) are worth a section of this chapter.

Date	**Fact**
1800	Alessandro Volta invented the first electrical battery capable to provide a continuous current supply to a system. The battery was made of pairs of copper and zinc discs separated by a layer of cardboard soaked in brine, used as electrolyte.
1836	John Frederic Daniel invented the Daniel cell as a solution to some problems found in Volta's battery. One of them was the hydrogen bubble generation that caused short battery life. The solution was to add a second electrolyte to consume the hydrogen.
1837–1860s	Improvements to Daniel's cell made by a number of scientists led to highly reliable batteries such that were implemented for supplying electrical energy in the American and British telegraph networks.
1859	Gaston Planté invented the first rechargeable battery, named lead-acid. This battery was built of a lead anode and a lead-dioxide cathode immersed in sulphuric acid.
1886	Carl Gassner patented the first known dry cell. It was known as a dry cell because it did not have a free liquid electrolyte; rather this liquid was combined with plaster of Paris to create a paste.
1896	Mass production of Gassner's dry cell by the National Carbon Company with a variation in the solid electrolyte. This type of cell is known as the zinc-carbon battery.
1899	Waldemar Jungner invented the rechargeable nickel-cadmium battery; it is better known as the alkaline battery because it used an alkaline electrolyte.
1903	Thomas Edison patented a rechargeable nickel-iron battery designed by Jungner.
1970	Assembly of non-rechargeable lithium cells used in watches,

	calculators and implantable medical devices.
1972–early 1990s	Development of suitable/safe combinations of anode, cathode and electrolyte to get rechargeable Lithium-ion batteries.
1989	Nickel-metal hydride batteries were introduced to the market. They are more environmental friendly than cadmium based batteries.
June 1991	Sony Corporation commercializes the recently discovered rechargeable lithium-ion battery with a nominal specific energy of 120-150 Wh/kg
1997	Lithium-ion polymer battery was developed. The electrolyte is a solid polymer composite instead of a liquid solvent. Another feature is that the electrodes and separators are laminated.
1997-2014	Lithium based batteries keep evolving and their specific energies getting higher.

Table 1. Historical evolution of battery technology, from [21], [22].

As can be seen from the analysis of Table 1, the current state of the art belongs to the lithium-based batteries. Current trends in research and development recognize this technology as the most promising as stated by published results of experiments done by several companies.

One of these published results was announced on February 27, 2012, by Envia systems; this company put under test a rechargeable lithium-ion battery cell obtaining a specific energy of 400 Wh/kg. The evaluation was performed by the Electrochemical Power Systems Department at the Naval Surface Warfare Center, sponsored by Advanced Research Projects Agency – Energy (ARPA-E). Even though this technology is fully tested and can supply an “enormous” amount of energy, it is not available in the market yet because is still transitioning to become an off the shelf product.

An example of how the specific energy of lithium-based batteries is growing rapidly is the fact that in October, 2013, A123 Venture Technologies started collaborating with Solid Energy Systems Corp. to integrate new lithium-ion battery technologies that could potentially deliver specific energies at levels up to 800 Wh/kg. These batteries are meant to operate safely in a temperature range of -40 to 250 degrees Celsius. Experimental results from prototypes are expected to be available by the end of 2014.

Battery technology is a field of science and technology that is constantly evolving, and it does not seem to have an ending in the near future. Many companies are expecting batteries with high specific energies that will enable greater capabilities of their products, such as portable devices or transportation vehicles. Figure 2 shows how specific energies have changed throughout the years of development and what the expectations for upcoming years are.

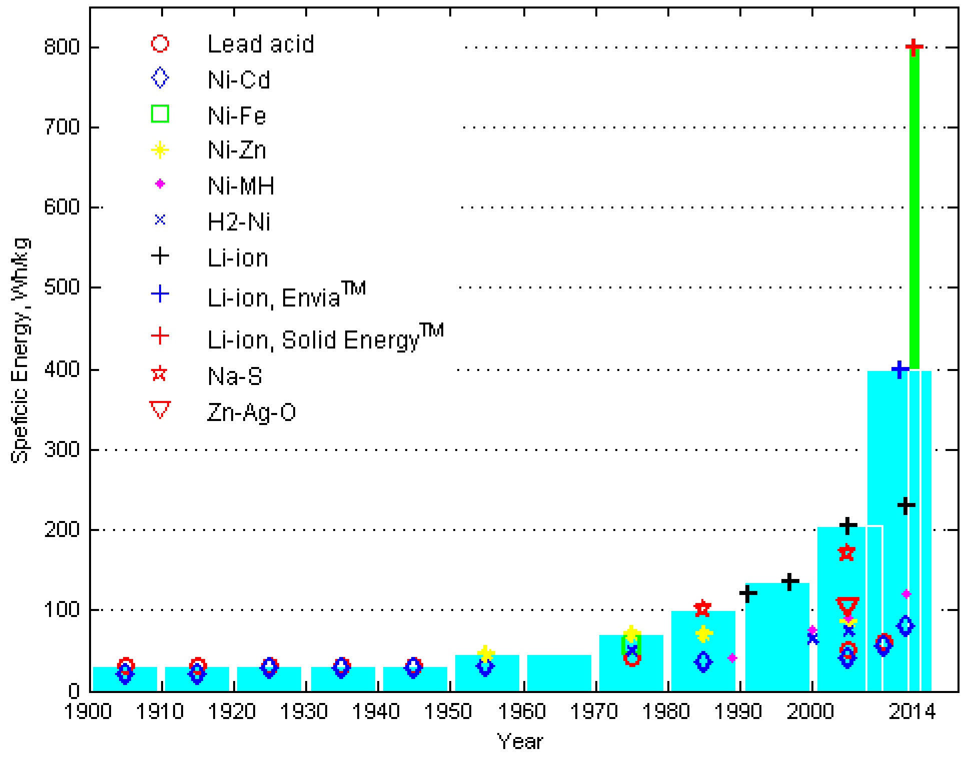


Figure 2. Evolution of specific energy for different battery chemistries, after [23].

7. Historical Review on Photovoltaic Cell Technology

As well as battery technology, PV cell technology has been evolving rapidly through the years. The main parameter to be improved in the PV cells is their efficiency, which is the percentage of the energy available from the solar radiation that can be transformed into electricity; however, there are also some other important parameters related to PV cells, such as their weight and rigidity. These parameters are inherited by the PV cells according to their chemical composition and manufacturing process; therefore, analyzing the evolution of the composition with their corresponding improvement in efficiency will give us a sense of the trend in developing new PV cells. Table 2 shows a historical timeline that represents the creation and evolution of photovoltaic solar energy conversion technologies. Within the table, note the difference between the thin film PV cell technology and the conventional PV cell.

Date	Fact
1839	Alexandre Edmond Becquerel discovers the photovoltaic effect. His experiment was done by illuminating two electrodes with different types of light. Electrodes were coated with light-sensitive materials and the experiment was performed in a black box surrounded with an acid solution. It is observed that the electrical current increases with the intensity of the light.
1873	Willoughby Smith discovered the photo conductivity of the selenium.
1876	William Grylls Adams and Richard Day discovered that a selenium cell exposed to light produces electricity.
1894	Charles Fritts built a solar cell made of selenium and gold. The solar cell had an efficiency of 1%.
1905	Albert Einstein published a paper about the photoelectric effect. In this work, he treated the light energy as being transported in discrete quantized packets.
1918	Jan Czochralski found a method to grow mono crystalline silicon, which increased the efficiency of the solar cell, being higher than the 1% of its predecessors.
1954	Calvin Fuller, Gerald Pearson, and Daryl Chapin at Bell Labs discovered a new silicon cell. This cell was more efficient than the selenium cell and could supply electrical energy for small devices. The efficiency achieved was ~4%.
1956	Solar cells became available in the market, but the prices are too high for their wide spread.
1958	The Vanguard I satellite was launched. This was the first PV-powered satellite. The cells were specially developed for the U.S. Army and had an efficiency of 14%.
1959	Hoffman electronics developed and commercialized 10% efficiency photovoltaic cells.
1960	Hoffman electronics achieved 14% efficiency photovoltaic cells.
1980	First thin-film solar cell was made, it had 10% efficiency.

1981	Paul MacCready built The Solar Challenger aircraft. This was the first solar-powered airplane. It had 16,000 solar cells supplying ~ 3 kW of power.
1985	The University of South Wales achieved 20% efficiency in silicon cells.
1992	University of South Florida developed a 15.9% efficient thin-film photovoltaic cell.
1994	The National Renewable Energy Lab (NREL) built a conventional solar cell with 30% efficiency. The cell was made of gallium indium phosphide and gallium arsenide.
1999	NREL tested an 18.8% efficient thin-film photovoltaic cell.
2007	The University of Delaware obtained 42.8% efficiency in a conventional solar cell.

Table 2. Historical evolution of photovoltaic cell technology, from [24] [25].

Figure 3 shows the evolution of different types of PV cells and the most advanced stage in every design.

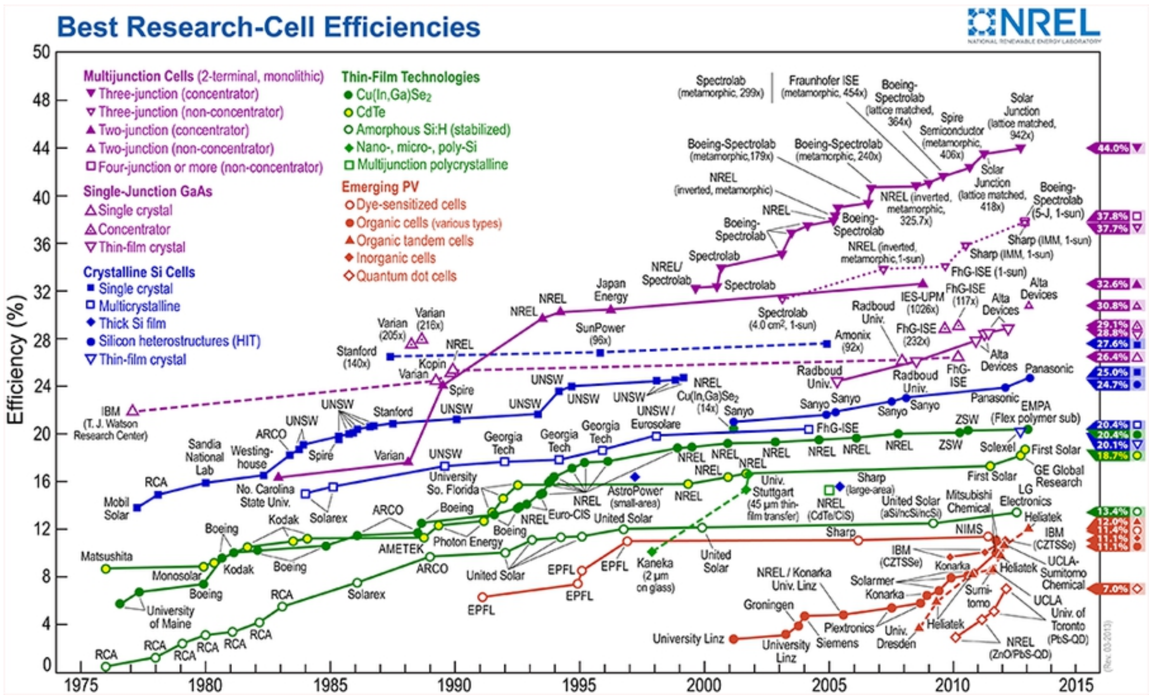


Figure 3. Evolution of photovoltaic (PV) cell technology, from [26].

8. Lithium-Ion/Polymer Battery Handling and PV Cell Behavior

After analyzing the state of the art in batteries and PV cells, it was decided that TaLEUAS will use lithium-based batteries. These batteries were chosen for their high specific energy and mono crystalline silicon thin-film PV cells due to their high efficiency and flexibility. There are safety considerations for handling the lithium-based battery cells given their unstable behavior when subjected to abnormal charging/discharging conditions. This behavior could lead to permanent damage to the batteries and the equipment where they are installed. Regarding the PV cells, the type of array—series or parallel—defines the behavior of the output power when the PV cells are shaded.

Most of the causes that could result in the failure of a lithium-ion/polymer battery are related to excessive heat generated during the charge/discharge working cycle. Heat released in the battery when in the charging/discharging condition is produced by the flow of electrons and the internal resistance. Therefore, temperature control is important to avoid reaching a temperature at which decomposition reactions in the electrode and electrolytes occur. This decomposition is seen as exothermic reactions; i.e., the higher the temperature, the stronger the reaction. On the other hand, the higher the charge/discharge rate, the higher the temperature generated.

Motivated by the safety considerations in handling of lithium-ion batteries, in 2012, Doughty and Roth [27] performed a study on safety in handling and operating lithium-Ion batteries. Outcomes of this study resulted in the design of a number of protective devices, including the shutdown separators, cell vent tabs, current interrupters, temperature protectors, current limiters, diodes, and advanced battery management systems. All these protective devices prevent temperatures from reaching the point where decomposition reactions begin. Batteries managed in accordance with the devised rules and conditions do not show failure modes, such as physical damage, thermal abuse due to excess of current, over charge/over discharge, and short circuit.

In addition to the use of protective devices, knowing in advance how a battery behaves by means of mathematical models allows the prediction of performance/failure of existing cells. Such mathematical models are based in physics, as well as electrochemical and thermodynamic phenomena that take place, and describe the behavior of the battery components during the charging and discharging processes. The mathematical model is also based on the properties of the specific feedstock (chemical materials) used to build the battery.

A very important parameter to be modeled is the variation of specific energy over time and charging cycles. The decrement in specific energy is due to a loss in capacity and an increment in the internal resistance of the battery. This is crucial because it determines the lifetime of a battery. This decrement in specific energy can be described as an aging phenomenon and depends mainly on the charging/discharging conditions. As most of the phenomena that occur in a battery, the process of decreasing in specific energy is also temperature dependent and is accelerated by high temperatures.

Early mathematical models were made by Sudoh and Newman [28] in 1990 for a sodium/iron chloride battery containing a molten $AlCl_3$-NaCl electrolyte. The outcomes of this research were the basis to predict some performance parameters in batteries, such as state of charge, cell temperature, and current-voltage relation during the charging/discharging cycles.

Even though the use of the most efficient photovoltaic cell is a critical mission defining parameter, the amount of light absorbed by the cell and the connecting architecture (series or parallel) are the most fundamental design parameters that define the efficiency of the PV array. Based on this, Ramabadran and Mathur in their work [29] in 2009 tested the effects produced on the delivered power by shading of series- and parallel-connected solar photovoltaic cells. Series connections are made to get the desired voltage out of the solar cells, and parallel connections are made to obtain a certain power. Results determined that parallel connections are less susceptible to shading than the connection in series.

E. CURRENT TECHNOLOGY

Potential advantages found nowadays in batteries and PV solutions will enable the technological and operational objectives of this thesis. Components are selected based on an assumption that keeping the maximum energy level onboard allows for more autonomy. The overall architecture (Figure 4) of the integrated design contains the following technological components that need to be neatly integrated onboard of a prototype glider platform:

- battery cells
- protection circuitry
- PV cells and maximum power point tracking (MPPT) unit
- low-power advanced autopilot, onboard computing unit and communication.
- composite materials and light weight structural components

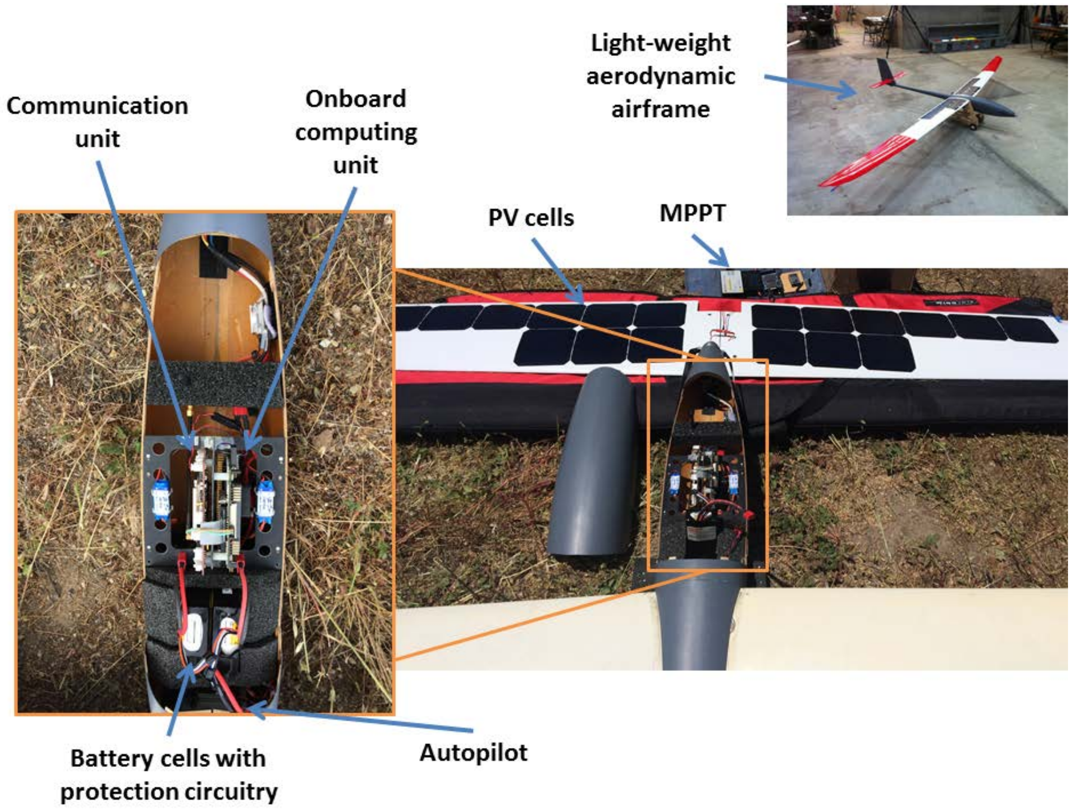

Light-weight aerodynamic airframe
Communication unit
Onboard computing unit
PV cells
MPPT
Battery cells with protection circuitry
Autopilot

Figure 4. Key hardware components integrated onboard TaLEUAS.

1. Battery Cell Technology

High specific energy storage technologies are rapidly evolving; thus producing more and more novel solutions every day. The objectives of this study will be accomplished using only available off-the-shelf components that are freely available on the market.

Rechargeable lithium-polymer battery cell technology will be the core of the Electrical Energy Management System (EEMS) of the UAV. Li-Po technology brings the following advantages:

- allow storing a significant amount of energy in a relatively small volume/mass – high specific energy
- offer a longer battery life compared with other technologies, given the same continuous charging/discharging condition

2. Protection Circuit Modules and Sensing Technology

To enable safe handling of lithium-polymer batteries it is necessary to control numerous variables that affect the state of charge of the system. This control process protects the battery from reaching the safety critical temperatures at which unstable decomposition reactions occur. Benefits of using these protective technologies are the following:

- Protection circuit modules prevent the batteries from failing by isolating them from the circuit when a malfunction occurs, such as: over charging, over discharging, high current, short circuit, and high temperature. This module also keeps the balance of charge among the battery cells in the same pack.
- Sensing of the internal characteristics (temperature, current, voltage) of each cell of a given battery pack allows monitoring of the actual status of the entire battery, which is given by voltage, current passing from or to the batteries, energy stored/remaining.
- When the batteries are integrated with the solar cells, the sensing capabilities should also be extended to account for the energy provided by the photovoltaic cells and the energy consumed by the onboard load.

3. Photovoltaic Cell and MPPT Technology

Power that is produced by the photovoltaic cells depends on a number of factors, such as the angle of incidence of solar cells toward the sun, shading of cells by the aircraft structures, and the efficiency of the solar cells. To account for a wide range of lighting conditions of an onboard solar array, an additional device called the MPPT is integrated into the energy management system to allow for maximum power supplied by the photovoltaic cell.

The core components of the solar powered glider include the high-efficiency, thin-film, mono crystalline PV cells and the MPPT unit. The integration of these two components allows efficient harvesting of solar radiation and transforming it into electricity. Moreover, some degree of structural flexibility of the mono crystalline PV cells allows for integration of the cells into the “skin” of the wings. Integrating the flexible cells conformal to the airfoil, results in an aerodynamically clean surface and thus does not penalize the design by inducing parasitic drag.

4. Low-Power Autopilots and Onboard Computer Technologies

These components are required to enable flight of the autonomous glider, and to run additional algorithms implemented for specific tasks or capabilities. Most of the commercially available autopilots do not have a complete solution suitable for objectives of the thesis. The autopilot and onboard computing microprocessor are the consumers of electrical energy because they are in service during the entire flight. Therefore, the lower their power consumption, the more extended the autonomy is.

5. Composite Materials and Manufacturing Technologies

Recent developments in materials science have revealed lighter and stronger composites. These composites together with novel manufacturing techniques result in lightweight and high aerodynamic efficiency airframes. Envisioned benefits of the combining of these technologies are as follows:

- embedding of thin-film photovoltaic cells in the airframe and wings without adversely affecting the structural and aerodynamic characteristics of the airplane
- less drag as a product of minimum roughness on the airframe's surface

F. THESIS OUTLINE

The rest of the thesis offers a logical and detailed explanation of how the energy extracted from thermals and the PV energy are combined to provide the extended endurance of a glider; each chapter contains the statement of the task to be solved, the theoretical approach and the methodology used for that purpose, and the results obtained from numerical simulations and experimental trials.

Chapter II discusses two approaches implemented in the detection of thermals. The first approach is based on online determination of the inherited sink rate of the glider, and the second one relies on the real-time estimation of the total energy with a Kalman Filter. The difference between the techniques and the benefits of both approaches are presented and discussed.

Once the thermal is detected, the thermalling guidance algorithms enable the glider with an autonomous soaring capability, which makes the glider climb along a self-centering path around the thermal. An existing thermal centering controller is modified with a combination of heuristic techniques. This new controller and the thermalling guidance are discussed in Chapter III.

The theory and mathematical apparatus describing the formation of thermals are explained in Chapter IV; some mathematical models are analyzed with respect to their fidelity in representing the characteristics of real thermals. Atmospheric and geographic conditions together with additional data are combined to determine the location of the thermals using Bayesian inference. This probabilistic approach is used by a team of autonomous gliders when they perform a cooperative search for thermals in a real operational scenario.

Chapter V presents the overall process of integrating the photovoltaic energy onboard with their key components; this process includes: solar panel embedding into the wings, Electrical Energy Management System (EEMS) design, and the contribution of the electric energy to the energy budget of the glider.

Results of flight tests with a single glider enabled with the developed algorithms are presented in Chapter VI. These results are used to validate the feasibility of the “solar-soaring” concept and the achievable performance of the single glider system. At the same time, these results were used to confirm the utility of high fidelity simulations provided by the simulation environment based on integration of Condor-Simulink.

Analysis of the experimental results and the corresponding conclusions are present in Chapter VII. Furthermore, some possible scenarios and applications feasible to the developed soaring UAVs are also presented here. A number of ideas and technologies that can be explored in future works conclude this chapter.

A number of appendices provide supplemental information about the project. In Appendix A, a description of the Condor-Simulink communication protocol is given with emphasis on the development of the Simulink decoding-encoding model that allows the MATLAB/Simulink development environment to communicate with the Condor soaring glider simulator. Some details highlighting the utility of the Condor high-fidelity soaring simulator are provided here for completeness.

Appendix B describes the process of embedding the solar cells into the wings of the autonomous glider.

Thorough details of the algorithms used on the EEMS developed to accurately measure the state of charge and the aging of the batteries are discussed in Appendix C. The EEMS primarily measures the amount of energy stored onboard the batteries, which determines the endurance of the autonomous glider. The code programmed to implement the complete algorithm running on the EEMS is presented in this appendix as well.

Appendix D contains the implementations materials (C-codes, m-scripts and Simulink models) that were developed in the MATLAB-Simulink development environment.

One paper resulting from this thesis has been published in the proceedings of the International Federation of Automatic Control (IFAC) conference. It is included into Appendix E.

II. DETECTION OF THERMALS

The natural upward motion of convective air can be used by a glider to climb and gain altitude. The interaction between the glider and the surrounding air is measured by the onboard inertial sensors. Real-time processing of the onboard measurements enables the detection and identification of the convective air updrafts. Two approaches to detect thermals are presented. The first approach compares the nominal sink rate of the glider with the currently measured sink rate. A difference between the inherent sink rate and the currently estimated one enables detecting of thermal activity. The second approach computes the total mechanical energy rate of a glider that tends to remain nearly constant in no-thermal conditions. Therefore, a variation of this total energy rate enables detection of the upward moving thermals by analyzing the sign of the energy derivative. Both approaches provide good results in detecting thermals in numerical simulations. Their performance is also tested in real flight tests with the results presented in Chapter VII.

A. OVERVIEW

In a piloted glider, an indication of the presence of a thermal in the atmosphere is first experienced by the pilot as a sudden force pushing the airframe upwards and therefore changing its sink rate. However, a better indicator is necessary to identify not only its presence, but its intensity. To accomplish these objectives, onboard variometers are used to help in the soaring task by giving audible or visual signals corresponding to the sink rate that the glider is experiencing. Among these devices, the Total Energy Compensated (TEK) variometer [30] gives the best measurement by also accounting for the changes in kinetic and potential energy. In a typical scenario, the pilot hears/watches what the change in sink rate is and then adjusts the glider bank angle to latch in the thermal and keep climbing by using the updraft flow of the thermal. Even though an audible-visual signal is useful in a piloted glider, translating this into a continuous-numerical signal—which is the type of signal that the autonomous glider requires—is not always possible or accurate.

B. METHODS

This section describes the mechanism that enables the autonomous glider with the thermal detection capability. There are two techniques that solve the detection task. The first technique analyzes the behavior of the glider with respect to the moving air around it, and the second technique analyzes the full time derivative of the total energy—potential and kinetic. The results demonstrate that both techniques detect thermals reliably; therefore, their onboard implementation guarantees a very high probability of thermal detection.

1. Sink Rate Polar

This approach relies on the assumption that while in flight the glider has a nominal behavior that relates the descending speed $\dot{h}$ (sink rate) with its True Air Speed (V_{TAS})—for every V_{TAS} measured in flight there is only one corresponding value of descending speed $\dot{h}$. This *sink rate* is different and specific for every configuration of the glider mechanization; the mechanization includes the number and specific settings of the control surfaces. This inherent property of the glider in each configuration can be experimentally measured in wings-level flight and no-wind condition. Since the $\dot{h}$ is so specific and can be known ahead of time, it is therefore possible to use the vertical velocity measured in flight to identify the sign of deviation from the expected inherent sink rate that in turn leads to the detection of the updraft.

The Sink Rate Polar is a plot that depicts on the "x" axis V_{TAS}, and on the "y" axis, the inherent sink rate $\dot{h}$. Using the Sink Rate Polar, it is possible to compute the maximum distance that could be traveled and the total loss in altitude in an interval of time. Other additional information which can be extracted from the plot is the optimal commanded speed used to fly the maximum distance with the minimum sink rate.

Even though the Sink Rate Polar is obtained at the design phase of an airplane, any small change in the lift/drag ratio due to a different configuration of the airframe, surface roughness, or change in weight leads to a different Sink Rate Polar. Therefore, it is desired to "generate" or learn this function for any possible configuration of the glider without resorting to the time-consuming analysis of wind tunnel experimentation.

Fundamental fluid dynamics of a typical aerodynamically clean airframe, such as a glider, suggests that the Sink Rate Polar can be represented by a second order polynomial; as described by Reichmann [13], the sink rate and the V_{TAS} are related in accordance with Equation (2.1).

$$\dot{h} = A \times V_{TAS}^2 + B \times V_{TAS} + C \qquad (2.1)$$

where $\dot{h}$ represents the sink rate, V_{TAS} is the True Air Speed, and A, B, C are the coefficients of the second order polynomial that describes the inherent Sink Rate Polar of the glider.

Parameters A, B, C in Equation (2.1) can be identified by applying a number of methods. The desire to have them identified during the real flight of a particular glider suggested the use of recursive linear least squares algorithm [31]. Therefore, a number of experiments have been performed to determine the Sink Rate Polar of gliders available at NPS; gliders have fixed but initially unknown characteristics of Sink Rate Polar. To verify the correctness of the online identification of the sink rate polar, a comparison of the obtained results was made with the results of Edwards [32] obtained in 2007 on the same glider airframe (SB-XC). Performing those experiments requires a lot of time to get a good set of samples within a range of speeds that spans from just above the stall speed to a considerably higher speed that does not compromise the structural integrity of the glider. Furthermore, the experiments must be performed in the conditions that exclude the influence of wind that is very difficult to control in an open environment. Overall the constraints include:

- airplane is in wings-level flight
- zero wind condition
- no thermals in the flight path

Repeatability is crucial in any experimentation; however, in real flight tests it is not possible to completely reproduce the experimental conditions; therefore, following the same procedure could lead to variable outcomes.

Consequently, this thesis considers the application of a recursive estimation algorithm that learns and identifies the Sink Rate Polar characteristics of the glider while in flight. The algorithm developed not only gives a fast converging response but also provides some extra benefits that include:

- the best approximation based on statistical averaging that excludes the contribution of the long and short period dynamics, rejects measurement noise and erroneous samples (dropouts are treated as disturbances)
- processes the data in real-time as the measurements come from the sensors, providing a continuous contribution of the new data
- computationally feasible, i.e. saves computational resources while giving the best result

The estimation algorithm selected is the Recursive Linear Least Squares (RLLS) algorithm. It is applied to identify the parameters of a system in real time. It gives the same curve fit polynomial as the least squares method (offline), but it "adapts" to the change in the inputs, thus adjusting the result at every instant of time as the new data comes.

A necessary condition to implement such an estimator is the linear relationship between the underlying structure of the model and the uncertain parameters. As such, the Equation (2.1) is transformed in its vector form as shown in Equation (2.2).

$$\dot{h} = A \times V_{TAS}^2 + B \times V_{TAS} + C = \begin{bmatrix} V_{TAS}^2 & V_{TAS} & 1 \end{bmatrix} \begin{bmatrix} A \\ B \\ C \end{bmatrix} \qquad (2.2)$$

To produce the best estimate, the RLLS algorithm assumes that the measurements have an error with zero mean Gaussian distribution. The inputs to the RLLS estimator at every instant are given by the mathematical model expressed for the Equations (2.3) and (2.4).

$$y(t) = \varphi_1(t)\theta_1^0 + \varphi_2(t)\theta_2^0 + \ldots + \varphi_n(t)\theta_n^0 = \varphi^T(t)\theta^0 \tag{2.3}$$

$$y(t) = \begin{bmatrix} \varphi_1(t) & \varphi_2(t) & \ldots & \varphi_n(t) \end{bmatrix} \begin{bmatrix} \theta_1^0 \\ \theta_2^0 \\ \ldots \\ \theta_n^0 \end{bmatrix} = \dot{h}(t) = \begin{bmatrix} V_{TAS}^2(t) & V_{TAS}(t) & 1 \end{bmatrix} \begin{bmatrix} A \\ B \\ C \end{bmatrix} \tag{2.4}$$

where y is an observed (measured) variable or state ($\dot{h}$), t represents the time step, $\varphi^T(t)$ are the known (also measured) functions (V_{TAS}) called regressors, and θ^0 are the unknown parameters of the model to be determined.

Using the relationship of the measurements as expressed in Equation (2.4), the RLLS algorithm (see Åström [31]) is computed recursively at every time step using Equations (2.5), (2.6), and (2.7).

$$K(t) = P(t-1)\varphi(t)(I + \varphi^T(t)P(t-1)\varphi(t))^{-1} \tag{2.5}$$

$$\theta(t) = \theta(t-1) + K(t)\left(y(t) - \varphi^T(t)\theta(t-1)\right) \tag{2.6}$$

$$P(t) = \left(I - K(t)\varphi^T(t)\right)P(t-1) \tag{2.7}$$

This recursive estimator is initialized with $P(t_0) = \left(\varphi^T(t_0)\varphi(t_0)\right)^{-1}$ using the first set of measurements and also providing the best guess for the initial value of the parameters $\theta(t_0)$ to be determined.

In the offline version of the least squares method, a sufficient condition to get the unique/best estimate of the parameters is that $\Phi^T\Phi$ must be invertible. This condition is called the excitation condition.

In a similar way, there is a condition that applies to the input signal *u* of the RLLS algorithm. This is called the persistence of excitation (PE) condition. The signal *u* is persistently excited if the matrix C_n in Equation (2.8) is positive definite and the limits of Equation (2.9) exist.

$$C_n = \lim_{t\to\infty}\frac{1}{t}\Phi^T\Phi = \begin{bmatrix} c(0) & c(1) & c(n-1) \\ c(1) & c(0) & c(n-2) \\ c(n-1) & c(n-2) & c(0) \end{bmatrix} \tag{2.8}$$

where:

$$c(k) = \lim_{t\to\infty}\frac{1}{t}\sum_{i=1}^{t} u(i)u(i-k) \tag{2.9}$$

The persistence of excitation condition gives the guideline to determine when the parameters estimated by the RLLS algorithm have converged to their true value.

Experiment Setup for Software in the Loop (SIL)

To verify the validity and the convergence properties of the RLLS algorithm before its onboard implementation, the algorithm in Equations (2.5), (2.6), and (2.7) is implemented in a Simulink model (Figure 5). The resulting code is used to identify the known Sink Rate Polar of an ASW27 glider given by the software simulation environment of the Condor simulator (see more details in Appendix A). The ASW-27 glider aerodynamics has been researched by Boermans et al. [3] by utilizing the wind tunnel data.

Assuming that the Condor simulator provides a fully controlled environment in terms of flight conditions, allows one to perform the system identification experiment with no wind present—the ideal but never occurring conditions for a real flight experiment. Telemetry from Condor is read and processed by the RLLS algorithm. The output of the RLLS method is the set of *A, B,* and *C* coefficients for the second order polynomial, see Equation (2.4).

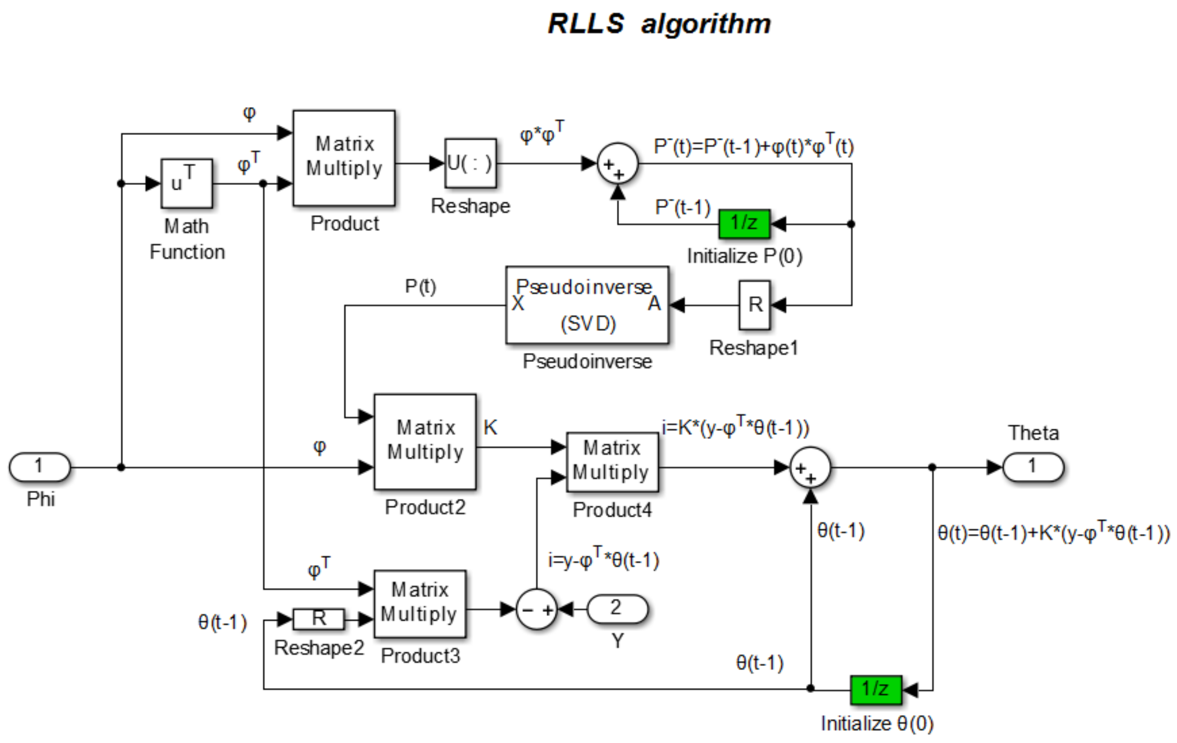


Figure 5. Simulink implementation of the RLLS algorithm.

The RLLS algorithm runs with the airplane performing a stable propulsion-free flight with no feedback control inputs present—open-loop experiment; the inherent longitudinal stability of the modeled glider and the long period diverging dynamics of lateral channel allow sufficient time for the data acquisition. The control surface that allows sampling the sink rate at different values of true airspeed is the elevator that is deflected in increments. Conceptually the experiment consists of the following. With the airplane in wings-level flight and traveling above stall speed, the elevator is moved in small increments allowing long enough time for the airplane to pass the short-period oscillations produced by its inherent motion, and finally, to settle down and recover the wings-level flight. After the transient is passed and a new set point of airspeed is achieved, a new sample of sink rate is taken. The experiment is repeated until the desired maximum true airspeed is achieved.

2. Total Energy Estimation

The total energy approach estimates the complete mechanical energy and the rate of variations of an airplane. The mechanical energy is composed of the potential energy (energy due to the altitude), the kinetic energy (energy due to the motion of the airplane), the loss of energy caused by parasitic drag (this term includes the skin friction drag and the pressure drag), and the energy loss produced by induced drag due to the lift generated by the glider. Mathematically, the total energy is expressed in inertial frame as shown in Equation (2.10).

$$E_{mec} = Kinetic + Potential + E_{drag} = \frac{mV^2}{2} + mgh - Dd \qquad (2.10)$$

where m is the mass of the glider, V is the inertial speed, D is the total drag force (parasitic plus induced), d is the distance along which that the drag force has been applied, and g is the gravity constant.

Consider an ideal scenario where an airplane is gliding in no-wind conditions at a certain altitude and speed with the engine/propeller turned off (no propulsion); the mass of the aircraft is assumed constant. Naturally, the glider constantly loses altitude due to the effects of the drag forces; hence, it loses energy. In order to avoid dependency on the mass of the airplane, normalize Equation (2.10) by $"mg"$, which results in Equation (2.11).

$$E_{norm} = \frac{E_{mec}}{mg} = \frac{Kinetic + Potential + E_{drag}}{mg} = \frac{V^2}{2g} + h - \frac{Dd}{mg} = \text{constant} \qquad (2.11)$$

In reality, the cumulative drag “D” can hardly be measured. Consequently, even though the energy estimated will only be the sum of kinetic plus potential, the change in the total energy is directly proportional to the energy loss due to drag, which can be observed in Equation (2.11).

According to the principle of conservation of energy, the sum of all sources and sinks of energy must be constant; hence, its variation (first derivative) over time is zero. To further illustrate the implications, let us obtain the first and second derivatives of equation (2.11) with respect to time.

$$\dot{E}_{norm} = \frac{V \cdot \dot{V}}{g} + \dot{h} - \frac{\dot{D} \cdot d + D \cdot \dot{d}}{mg} = \frac{V \cdot \dot{V}}{g} + \dot{h} - \frac{\dot{D} \cdot d + D \cdot V}{mg} = 0 \qquad (2.12)$$

$$\ddot{E}_{norm} = \frac{\dot{V}^2 + V \cdot \ddot{V}}{g} + \ddot{h} - \frac{\ddot{D} \cdot d + 2\dot{D} \cdot V + D \cdot \dot{V}}{mg} = 0 \qquad (2.13)$$

Note, that states V, $\dot{V}$, and $\ddot{V}$ can be either directly measured or estimated; on the other hand, D cannot be measured directly. At the same time, an aerodynamically clean glider with high glide ratio flying at constant speed ($\dot{V} \approx 0$) is expected to have a very shallow drag versus airspeed dependency, which suggests $\dot{D} \approx 0$. Therefore, the variation of energy of an aerodynamically efficient glider can be approximated by the following:

$$mg \cdot \dot{h} = D \cdot V \qquad (2.14)$$

$$\ddot{E}_{norm} = \ddot{h} \qquad (2.15)$$

The first equation allows for precise characterization of the total drag although it can hardly be accomplished by direct sampling of states in flight. Furthermore, in the idealistic descent a glider has $\dot{h}$ negative; therefore, at constant speed $\dot{E}$ is negative and its variation depends primarily on the change of the altitude. Now the changes of sign of $\dot{E}$ can be used to detect the thermal updraft. When there is a convective thermal that makes $\dot{h}$ positive, the thermal exists.

The total energy approach has been previously used by a number of researchers to detect convective thermals; see, for example, Allen and Lin [9] and also Andersson et al. [12]. Their heuristics-based approach and implementation of the energy-based detection algorithms feature a considerable lag time that can potentially lead to a failure of detecting a thermal.

The approach adapted in this project is a classical Kalman Filter (KF) that is formulated to estimate V, h, and their two derivatives. As soon as these states become available, their values are used to calculate the total energy E and its two derivatives $\dot{E}$, and $\ddot{E}$. Linear KF is applied to linear dynamics of V and h; it provides the best estimates of those states in the presence of Gaussian noise in measurements; the algorithm has fast speed of convergence thus significantly reducing the inherent delay in the energy estimation given by previous researches (see Allen and Lin [9] and Andersson et al. [12]).

As stated before, the drag D is not measurable; therefore, energy equations will be constructed with the available inertial measurements from the onboard sensors, such as the speed V and the altitude h; these equations are expressed in (2.16).

$$\begin{aligned} E &= \frac{V^2}{2g} + h \\ \dot{E} &= \frac{V \cdot \dot{V}}{g} + \dot{h} \\ \ddot{E} &= \frac{\dot{V}^2 + V \cdot \ddot{V}}{g} + \ddot{h} \end{aligned} \tag{2.16}$$

These equations describe the normalized potential and kinetic energies and their derivatives with respect to time. From Equation (2.16) it can be observed that not only the speed V and the altitude h are required but also their first two time derivatives.

It is common to find certain levels of noise in the measurements coming from sensors installed onboard; therefore, a KF is implemented to eliminate such noise, while estimating the values of the time derivatives of V and h. The relationship among these variables is given by the Equation (2.17).

$$\begin{aligned} h_t &= h_{t-1} + \dot{h}_{t-1}\Delta t + \frac{\ddot{h}_{t-1}\Delta t^2}{2} \\ \dot{h}_t &= \dot{h}_{t-1} + \ddot{h}_{t-1}\Delta t \\ \ddot{h}_t &= \ddot{h}_{t-1} \\ \\ V_t &= V_{t-1} + \dot{V}_{t-1}\Delta t + \frac{\ddot{V}_{t-1}\Delta t^2}{2} \\ \dot{V}_t &= \dot{V}_{t-1} + \ddot{V}_{t-1}\Delta t \\ \ddot{V}_t &= \ddot{V}_{t-1} \end{aligned} \tag{2.17}$$

where t represents time, Δt is the sampling time, V is the inertial speed, and h is the altitude measured in the inertial frame. Equation (2.17) describes the state transition model for the KF that will be implemented in Equations (2.18).

$$\hat{x}_t^- = A\hat{x}_{t-1} + Bu_t \tag{2.18.1}$$

$$\hat{\in}_t^- = A\,\hat{\in}_{t-1}\,A^T + \sigma_x \tag{2.18.2}$$

$$K_t = \hat{\in}_t^-\,C^T\left(C\,\hat{\in}_t^-\,C^T + \sigma_z\right)^{-1} \tag{2.18.3}$$

$$\hat{x}_t^+ = \hat{x}_t^- + K_t\left(z_t - C\hat{x}_t^-\right) \tag{2.18.4}$$

$$\hat{\in}_t^+ = \left(I - K_t C\right)\hat{\in}_t^- \tag{2.18.5}$$

where t represents the time steps, the - and + signs denote the estimates at the current time step before and after being updated respectively, A and B determine the state transition model, C is the measurements matrix, u is the control input to the process (zero for our case), $\hat{x}$ are the predicted states, σ_x is the variance matrix for the state prediction, $\hat{\in}$ is the predicted variance matrix, σ_z is the variance matrix in measurements, K is the Kalman gain, and z is the vector containing the sensor readings. Such readings are available coming from different sensors, which improves the convergence in estimating the states and eliminating the noise; V and h are taken from GPS measurements, while their derivatives are obtained by rotating the acceleration measured in body frame by the Euler angles-based Direction Cosine Matrix (DCM). The content of every term shown in Equation (2.18) is expressed in Equation (2.19).

$$\hat{x} = \begin{bmatrix} \hat{h} \\ \dot{\hat{h}} \\ \ddot{\hat{h}} \\ \hat{V} \\ \dot{\hat{V}} \\ \ddot{\hat{V}} \end{bmatrix}, \quad A = \begin{bmatrix} 1 & \Delta t & \frac{\Delta t^2}{2} & 0 & 0 & 0 \\ 0 & 1 & \Delta t & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \Delta t & \frac{\Delta t^2}{2} \\ 0 & 0 & 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \qquad (2.19)$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix},$$

$$\sigma_x = \begin{bmatrix} \sigma^2_{h_{process}} & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma^2_{\dot{h}_{process}} & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma^2_{\ddot{h}_{process}} & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma^2_{V_{process}} & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma^2_{\dot{V}_{process}} & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma^2_{\ddot{V}_{process}} \end{bmatrix}, \quad z = \begin{bmatrix} h \\ \dot{h} \\ \ddot{h} \\ V \\ \dot{V} \end{bmatrix}$$

and $$\sigma_z = \begin{bmatrix} \sigma^2_{h_{measurement}} & 0 & 0 & 0 & 0 \\ 0 & \sigma^2_{\dot{h}_{measurement}} & 0 & 0 & 0 \\ 0 & 0 & \sigma^2_{\ddot{h}_{measurement}} & 0 & 0 \\ 0 & 0 & 0 & \sigma^2_{V_{measurement}} & 0 \\ 0 & 0 & 0 & 0 & \sigma^2_{\dot{V}_{measurement}} \end{bmatrix}$$

The algorithm—the set of equations (2.18)—is initialized at $t = 0$ with some previous knowledge of the state values ($\hat{x}_0$) and the predicted variance matrix ($\hat{\in}_0$). These values will determine how fast the KF will converge to the estimated states after initialization is done.

$$\hat{x}_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \text{ or } \hat{x}_0 = \begin{bmatrix} h_0 \\ \dot{h}_0 \\ \ddot{h}_0 \\ V_0 \\ \dot{V}_0 \\ 0 \end{bmatrix}, \qquad \hat{\in}_0 = \sigma_x \tag{2.20}$$

Notice that the process model is linear time-invariant; therefore, at some moment, the Kalman gain (K) converges under observability assumption; consequently, the algorithm described in Equation (2.18) is simplified without the need of computing this gain (K) in real-time; it is just necessary to compute it once off-line using the discrete time Riccatti Equation.

Experiment Setup for SIL

The KF is implemented in the Simulink (Figure 6) environment using MATLAB embedded functions (Appendix D). Even though it can be implemented using Simulink blocks, the use of functions gives the user a better understanding of how the algorithm works.

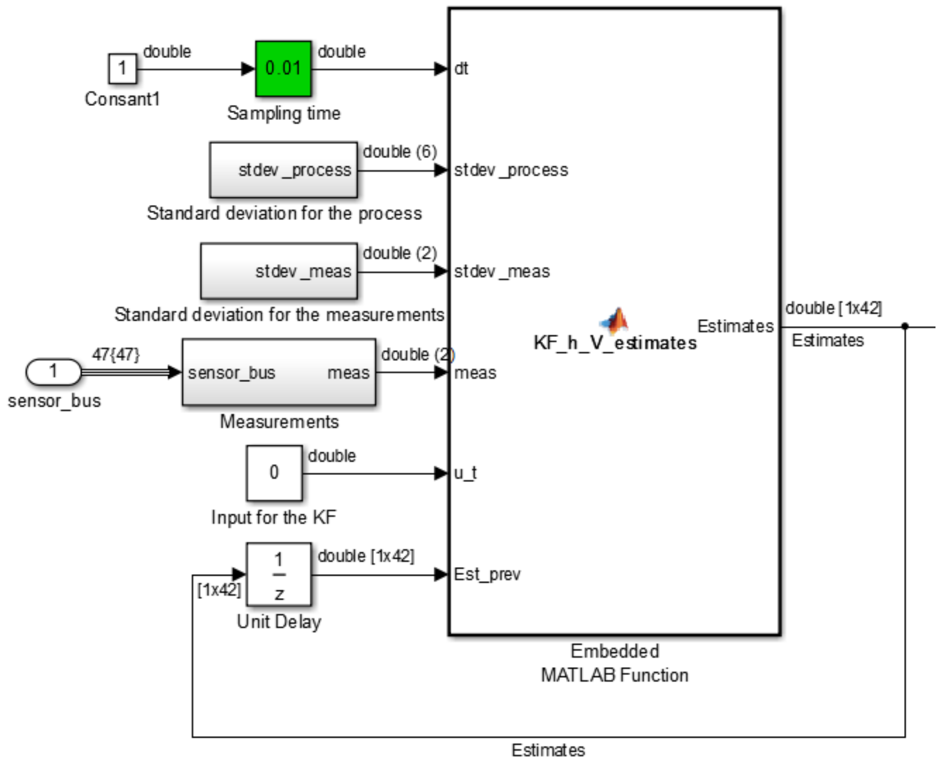


Figure 6. Simulink implementation of the Kalman Filter equations.

In the Simulink model, the signal "Estimates" contains the state vector and the covariance matrix to be used in the next iteration. In order to test the KF, it is necessary to know the variances of the noise of the process and of the measurements, as well as some knowledge about the initial values of $\hat{x}_0$ and $\hat{\epsilon}_0$. Usually, the variances are the tuning knobs that tune the quality of noise rejection by calculating the optimal value of Kalman gain. Given that the Condor simulator provides relatively "clean" measurements with minimal noise, the values for the standard deviations of the process and measurements noise are assumed negligible. In a real flight test, however, these standard deviations will increase significantly due to the inaccuracy of real onboard sensors.

The verification of the performance properties of the developed KF solution is based on the telemetry data provided by the Condor simulator for the ASW-27 glider; the approach is similar to the one used in the identification of the sink rate polar.

C. RESULTS AND DISCUSSION

The result of implementing the RLLS algorithm to identify the sink rate polar of an ASW-27 glider in SIL is presented in Figure 7. The comparison of the Sink Rate Polar obtained by the RLLS in the Condor simulator with the data reported by Boermans et al. [3] illustrates high degree of approximating the real sink rate characteristics. It is evident that the simulation environment captures quite well the aerodynamic characteristics of the glider and its interaction with the atmospheric phenomena; however, a minor difference still exists between the two Sink Rate Polars. This difference might be caused by the differences in the glider mechanization, and possibly by some numerical inaccuracies of the implementation of aerodynamics in Condor.

Even though the small differences are present, the result gives a good sense of the performance and accuracy of the RLLS algorithm in identifying the Sink Rate Polar. The polar provides a set of characteristics that can be used at the mission planning phase. These characteristics include the speed for minimum sink "$V_{s\min}$", the stall speed to be avoided, and the optimal gliding speed for maximum distance "V_{cc}." The latter one gives the best ratio of the sink rate and the gliding speed that is used in cross-country maximum distance flight of gliders.

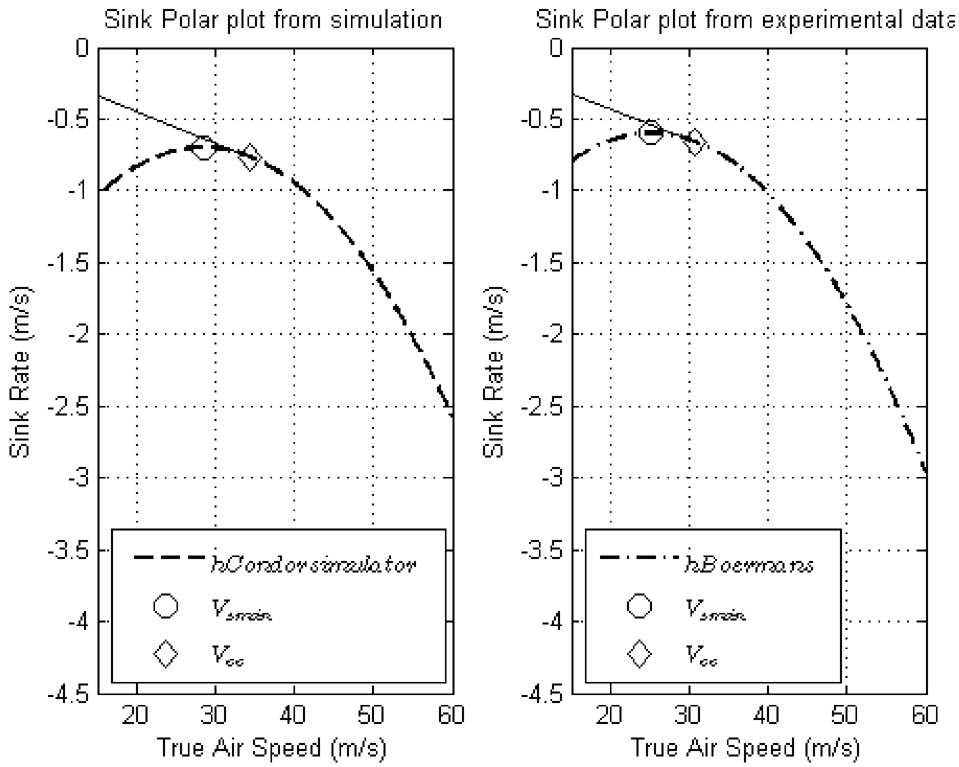


Figure 7. Identification of Sink Rate Polar in SIL.

Identified Sink Rate Polar provides direct detection mechanism of thermal updrafts; i.e., if the currently measured sink rate at a specific airspeed is smaller than the one predicted by the sink rate polar, then the updraft exists. Furthermore, if the sink rate is strong enough (that is, the difference between the actual measurement and the prediction is big) to allow the glider to gain altitude, then the effective thermal is present and the thermalling guidance law needs to be activated. The strength of the updraft is specified by a threshold corresponding to the minimum desired vertical rate that the glider needs to climb; for the software simulations of this study it was chosen at 1m/s.

To illustrate the efficiency of the total energy estimation approach in detecting thermals, a scenario is chosen where a glider needs to fly through the core of a thermal in the Condor simulator. The thermal is detected based on the change in $\dot{E}$. To verify both approaches in thermal detection, the outputs of the sink rate and the energy based algorithms are compared with an output of the TEK variometer supplied by the Condor. The results of detecting of thermals provided by two algorithms are very similar; additional tuning of the thermal strength parameter in the sink rate approach has a potential of making the detection simultaneous by both algorithms. The details corresponding to the energy approach are provided in Figure 8.

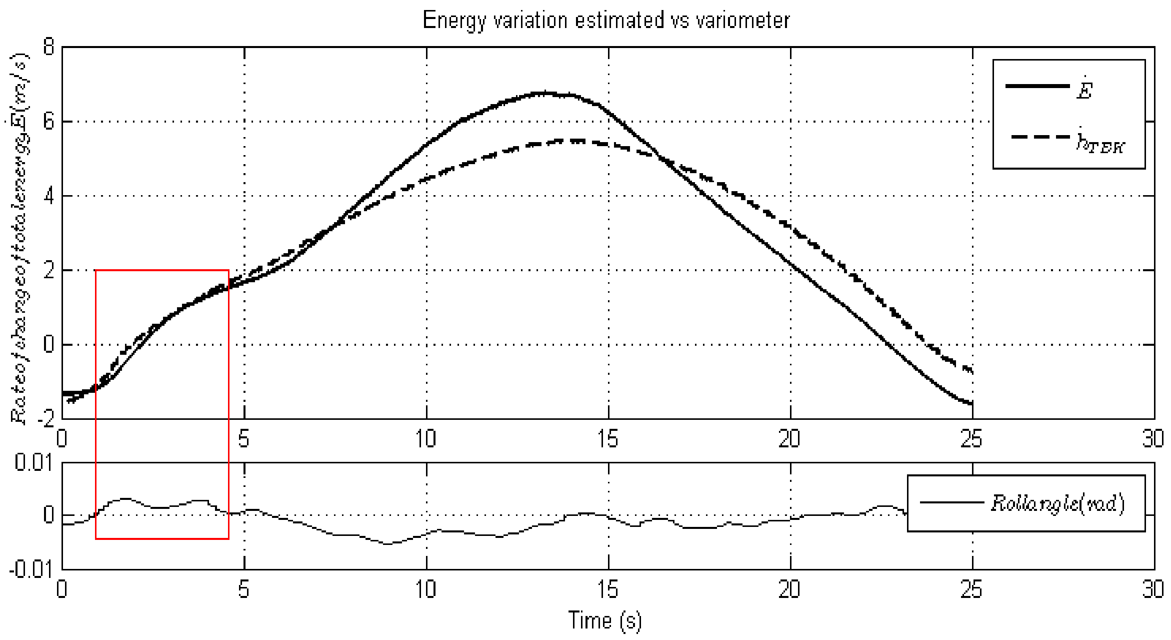


Figure 8. Characterization of a thermal based on the energy variation $\dot{E}$ and its comparison with a simulated TEK variometer.

Figure 8 shows how the $\dot{E}$ variable changes as the glider passes through the thermal; at the beginning and at the end of the thermal $\dot{E} \approx -1.8 m/s$ which is a result of sink zone that always accompanies an updraft; nominally the inherent energy loss rate $\dot{E} > -1 m/s$ which corresponds to the glider flying at $30 m/s$. When the glider enters the thermal, a rapid increase in $\dot{E}$ can be observed; the value of $\dot{E}$ keeps increasing and reaches a peak value at the core of the thermal and then decreases. Depending on the characteristics of the glider, a threshold in the $\dot{E}$ value is chosen to determine the presence of a thermal. When the glider enters a thermal, the lift force is not uniformly distributed over the wingspan, that directly affects the roll angle ϕ. Since the attitude of the glider is measured onboard, it can be used to detect the relative position of the glider with respect to the core of the thermal. Consequently, analyzing ϕ and $\dot{E}$ makes possible the determination not only of the presence of a thermal, but also the desired direction of turn towards the center of the thermal.

In conclusion, the sink rate and the energy identification algorithms enable reliable detection of thermal updrafts, while also providing useful characterization of commanded airspeed essential in thermalling guidance and in cross-country flights.

III. THERMALLING GUIDANCE

A glider's climb rate when in thermalling mode has a direct relationship with the potential energy gained by the aircraft. Therefore, maximizing the climb rate optimizes the gain of potential energy. The best climb rate is determined by the thermalling guidance controller that runs onboard the glider. The thermalling controller must account for two basic criteria: stability of flight and performance of climb. This chapter analyzes the performance of an existing thermalling controller based on Reichmann's soaring technique [13], which is commonly used by human pilots for centering in thermals. The material also compares the Reichmann technique with a newly developed thermalling controller that combines the benefits of Reichmann's approach with the capability of climbing close to the core of the thermal—Jay's soaring technique [33]. As a result, the new feedback controller gives better performance by maximizing the climb rate.

A. OVERVIEW

The process of extracting energy from a thermal can be divided into three steps. The first step is the detection either by the Sink Rate or by the Total Energy approaches. The second step is to locate or estimate the geometry of the thermal by utilizing the onboard measurements of the glider. Finally, the third step consists of climbing efficiently while "staying" in the thermal—called thermalling guidance. Human pilots have succeeded in thermalling guidance by implementing several techniques such as the ones used by Reichmann [13] and Jay [33], which were developed based on their experience in soaring flight. In order to implement this soaring capability onboard an autonomous glider, it is necessary to "emulate" those techniques and translate them into numerical algorithms that would provide the desired performance. Andersson et al. [12] successfully developed a feedback centering controller, which causes the glider to center at the thermal's core; the approach that formalizes the heuristics of Reichmann's technique is used throughout the chapter as the basis to compare the performance of new algorithms designed in this chapter. One of the constraints of the controller is the use of tunable parameters, which, depending on the characteristics of different thermals, gives variable performance when compared to the performance of a human-piloted glider. Therefore, there is an obvious need to optimize the thermal centering guidance by designing new algorithms that will "learn" the best set of parameters corresponding to different shapes and strengths of the convective thermals.

The purpose of this chapter is to design and implement onboard a thermalling feedback controller that gives the best climb rate in the presence of various thermals with different characteristics.

B. METHODS

There are two thermalling techniques that are widely known and commonly used by pilots. The first technique is called the "tighten on the surge." The approach tracks the first derivative of the energy and evaluates when its value is the greatest. When it happens, the bank angle is increased to make a sharper turn—Jay's technique [33]. The second technique states that the glider should widen out when finding the strongest lift, hence reducing the angle of attack—Reichmann's technique [13]. Even though both techniques may seem contradictory, they are both correct, with the caveat that they offer advantages in different situations.

The "tighten on the surge" technique is applied right after a thermal is detected and it is desired to place the glider in the center of the thermal and to optimize its climb rate. The reasoning behind this technique relies on the fact that the closer the glider is to the core of the thermal, the better the climb rate will be. The Reichmann's technique should be used when a new thermal is likely to develop near the current one and it is desired to get into that new thermal quickly. The signs that indicate the possibility of transition include a noticeable decrement in the rate of climb, or when one side of the thermal seems stronger than the other.

The formal translation of Reichmann's technique into a mathematical model and further into a thermalling feedback control law is shown in Equation (3.1), which is adopted from Andersson's paper [12].

$$\dot{\psi}_c = \frac{V}{\rho_d} - k_1 \ddot{E} \tag{3.1}$$

where $\dot{\psi}_c$ is the heading rate command, V is the ground speed, ρ_d is the desired turn radius around the thermal, $k_1 > 0$ is a feedback gain, and $\ddot{E}$ is the second derivative of the total energy computed as presented in the previous chapter.

The control law (3.1) contains two terms. The first term gives the heading rate command in steady state—when there is no variation on the $\ddot{E}$ term—for the glider to turn in a circle of constant radius; and the second term provides a dynamic feedback term that is proportional to the second derivative of the energy, $\ddot{E}$. It was explained in Chapter II that the strength of a thermal is characterized by $\dot{E}$; therefore, analyzing $\ddot{E}$ integrates the strength of the thermal as a feedback measurement.

Andersson's controller was proven in real flight and demonstrated a good performance in centering thermals, but the climb rate obtained was not compared with some other algorithms. Therefore, using the capabilities of the Condor simulator, I tested the performance of Andersson's controller onboard an autonomous glider against a glider controlled by an experienced pilot. As a result of this, it was observed that the climb rate of the human-piloted glider was higher than that of the autonomous one. The main reason for this result is the human-piloted glider tried to be at the core of the thermal for a longer time.

As demonstrated by human pilots in cross-country competitions, Jay's technique might provide better climbing results; therefore, it is worthwhile to implement it as a control law. Jay's technique can be translated in a mathematical model as expressed in Equation (3.2).

$$\dot{\psi}_c = \frac{V}{\rho_d} + k_2 \dot{E} \tag{3.2}$$

where most of the terms are the same as in Equation (3.1) except $k_2 > 0$, which is a tunable feedback gain, and $\dot{E}$ (energy rate) is the strength of the thermal.

In contrast to Equation (3.1), Equation (3.2) guides the glider to the core of the thermal when a higher strength is detected, thus enabling the glider to stay there as long as possible; hence, it climbs faster. Furthermore, it is in fact possible to combine the core elements of both techniques. Reichmann's technique self corrects the glider flying along a circular path around the core of the thermal. The Jay's technique directs the glider to the core of the thermal. A control law that combines both techniques is given by the following equation:

$$\dot{\psi}_c = \frac{V}{\rho_d} - k_1 \ddot{E} + k_2 \dot{E} \qquad (3.3)$$

where the feedback control gains k_1 and k_2 are the same as before.

C. EXPERIMENT SETUP FOR SIL

In unmanned systems, it is common the implementation of states machines to determine the actions that such unmanned system will take under certain circumstances. Simulink allows implementing these state machines, which are the mathematical models of computation used to design sequential logic systems. Inside a state machine there are a finite number of states, and the machine can go through them when certain conditions are met, causing a transition between states. At every moment, the system can be in one of several states depending how the programmer designs the state machine. At every state it is possible to run any function or code from the MATLAB/Simulink environment. This allows for great versatility in controlling the states of a glider when in the thermalling guidance mode. In building the capabilities of the autonomous glider, the state machine is designed to accomplish several tasks:

- determine when all conditions are met to detect a thermal
- locate the thermal with respect to the glider
- establish the direction of turning of the glider
- keep the glider turning in that direction with the thermalling controller running

- evaluate if the glider is ascending or descending
- determine when there is no more energy available in the thermal
- disable the use of the thermalling controller when the glider reaches the airspace ceiling, when there is no energy available in the thermal, or when there is a misdetection of a thermal.

The state machine implemented in Simulink for the thermalling guidance is shown in Figure 9. In a nominal condition, the glider is in the state of “No_Thermal.” At every instant of time, the state machine analyzes the values of $\dot{E}$, $\ddot{E}$, and the bank angle (ϕ); when these variables reach certain values, the state machine detects a thermal and determines its position. As a result, the next state will be either “In_Left_Thermal” or “In_Right_Thermal.” To return to the “No_Thermal” state, the state machine checks whether the glider is descending, the strength of the thermal is too small, and the glider is flying over the ceiling of the airspace.

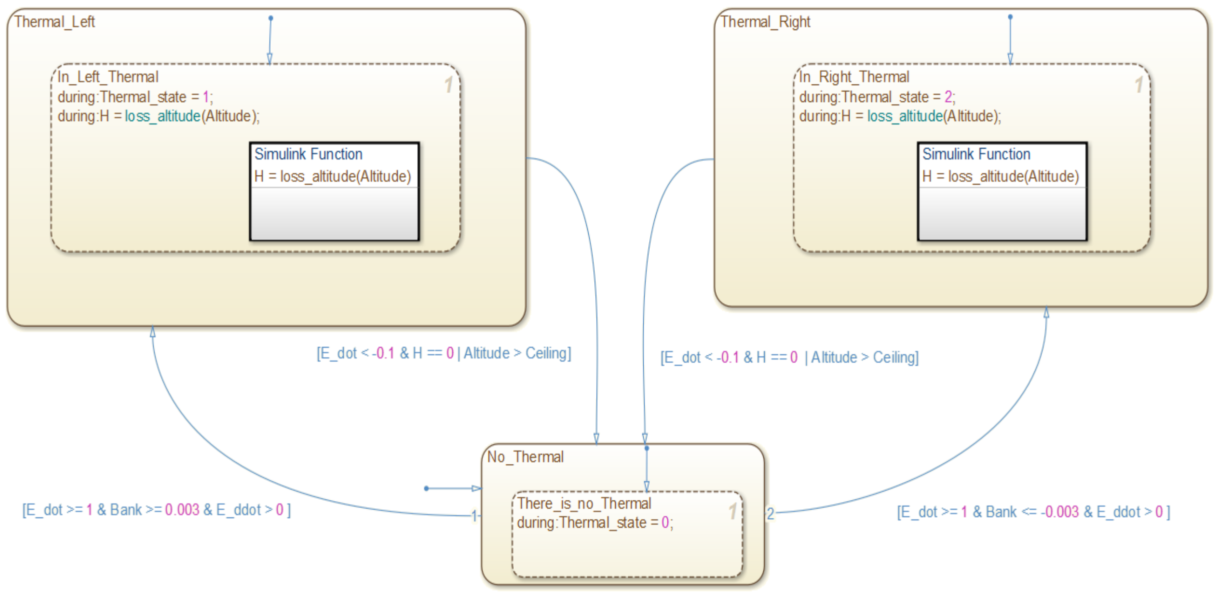

Thermal_Left
In_Left_Thermal
during:Thermal_state = 1;
during:H = loss_altitude(Altitude);
Simulink Function
H = loss_altitude(Altitude)
1
Thermal_Right
In_Right_Thermal
during:Thermal_state = 2;
during:H = loss_altitude(Altitude);
Simulink Function
H = loss_altitude(Altitude)
1
[E_dot < -0.1 & H == 0 | Altitude > Ceiling]
[E_dot < -0.1 & H == 0 | Altitude > Ceiling]
No_Thermal
There_is_no_Thermal
during:Thermal_state = 0;
1
[E_dot >= 1 & Bank >= 0.003 & E_ddot > 0]
1
2
[E_dot >= 1 & Bank <= -0.003 & E_ddot > 0]

Figure 9. State machine for the thermalling guidance implemented in Simulink.

Figure 10 outlines the implementation of Andersson’s centering controller in the Simulink environment.

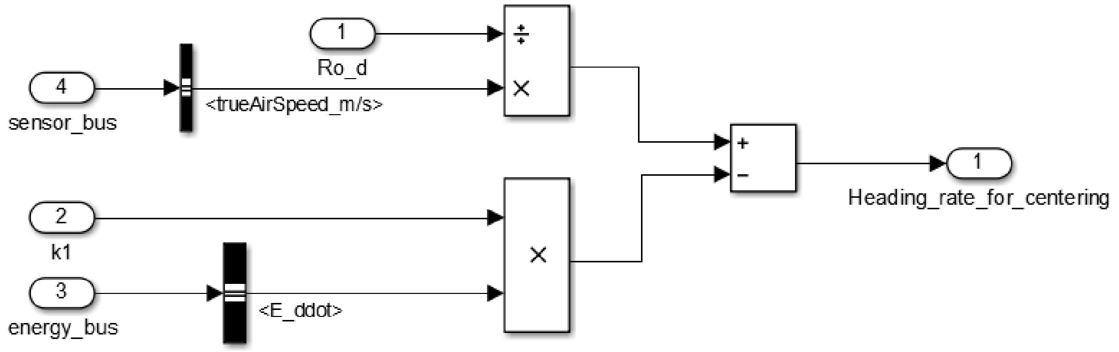


Figure 10. Andersson's centering controller implementation in Simulink.

There are two tunable parameters in Andersson's controller, which are selected based in some specific criteria. The minimum ρ_d is determined by the minimum operational turning radius of the glider. The control authority, that $\ddot{E}$ has over the guidance law, is dictated by k_1. Both parameters are bounded with upper and lower limits to guarantee stability of the controller as it was proven in [12].

Implementation of Equation (3.3), which combines the two thermalling techniques described previously, is outlined in Figure 11.

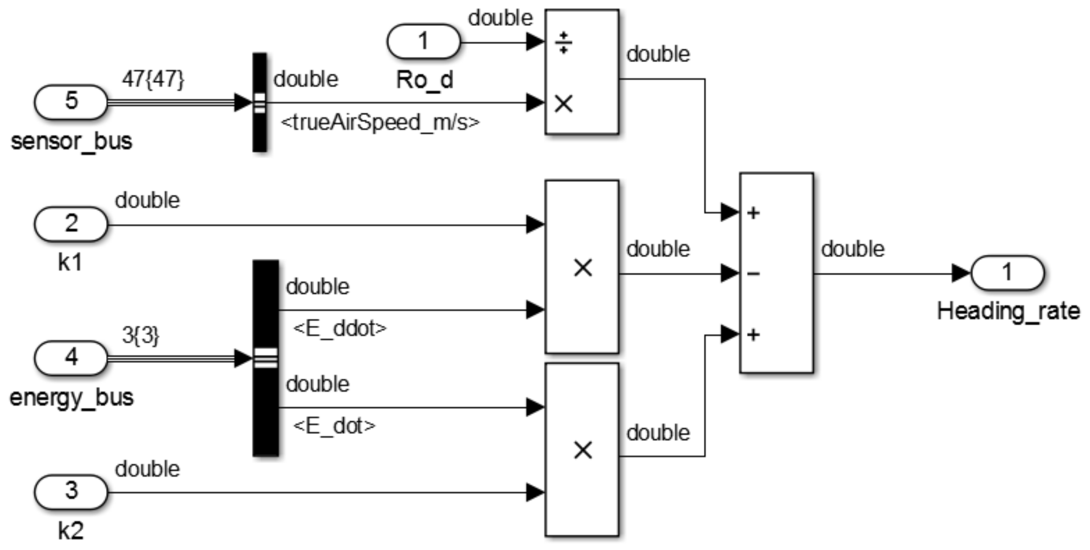


Figure 11. Implementation of thermalling controller of Equation (3.3).

I used the repeatability feature of the Condor simulator, which allowed me to generate the same atmospheric conditions—wind, location and strength of thermals, and sun incidence—every time the simulation was started. This feature was used to tune the gains of Andersson’s controller and the controller in (3.3) for a specific thermal. The parameters that provide the best climb rate for both controllers are as follows: for Andersson’s controller $\rho_d = 110 meters$, $V = 30 meters/second$, and $k_1 = 0.2$; and for the controller (3.3) $\rho_d = 110 meters$, $V = 30 meters/second$, $k_1 = 0.5$, and $k_2 = 0.02$. From the stability analysis, each gain value is selected to give a good performance without making the system unstable under typical flight conditions. After two controllers are properly tuned, a verification simulation is started with two identical ASW-27 gliders. Both gliders are set to fly to the same thermal that was used for tuning of their gains. The results are presented in the following section.

D. RESULTS AND DISCUSSION

The initial conditions for the encounter geometry—thermal and glider—are shown in Figure 12. Both gliders start the simulation at an altitude of 2000 m, both are equipped with the same state machine and different thermalling controllers; the simulation ran for 230 s, which is sufficient to compare their detection and climbing performance. When the thermal is detected, the thermalling guidance is activated to generate the best climb rate out of every controller. Figure 13 describes the paths of both gliders, from their initial position until the final altitude, when the simulation is stopped. It is observed that controller (3.3)—using the Reichmann and Jay techniques combined—converges faster at the center of the thermal and stays longer time in the core. On the other hand, Andersson’s controller—using only the Reichmann technique—converges more slowly to the center of the thermal and obtains a slower climb rate as well.

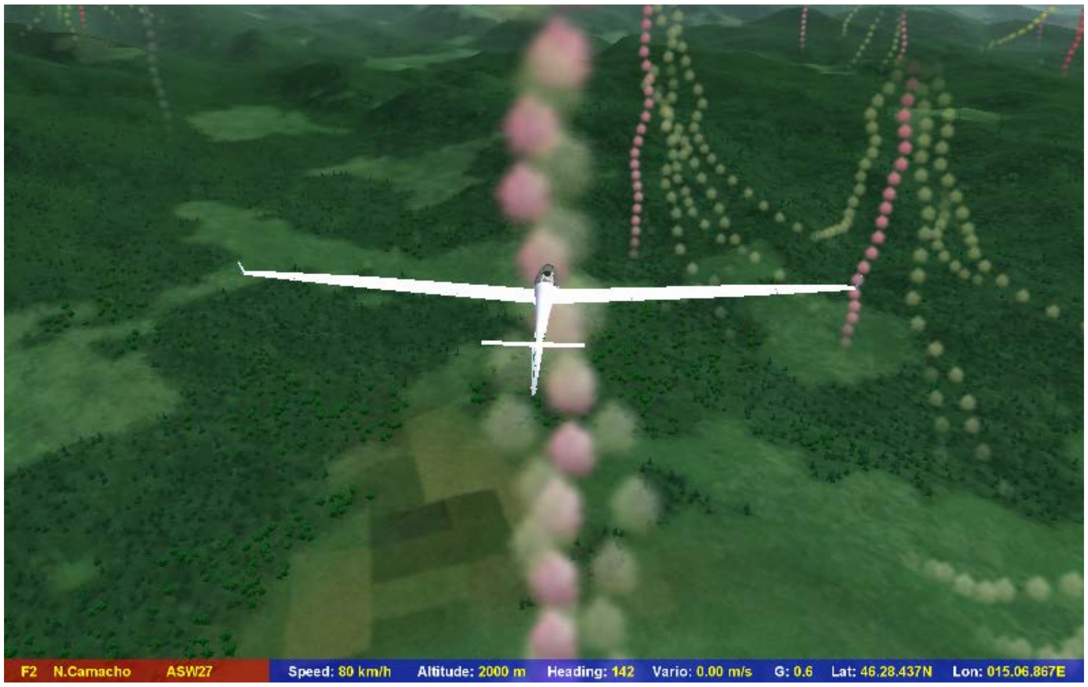

F2 N.Camacho ASW27
Speed: 80 km/h Altitude: 2000 m Heading: 142 Vario: 0.00 m/s G: 0.6 Lat: 46.28.437N Lon: 015.06.867E

Figure 12. Initial conditions for testing performance of Andersson's controller and Equation (3.3).

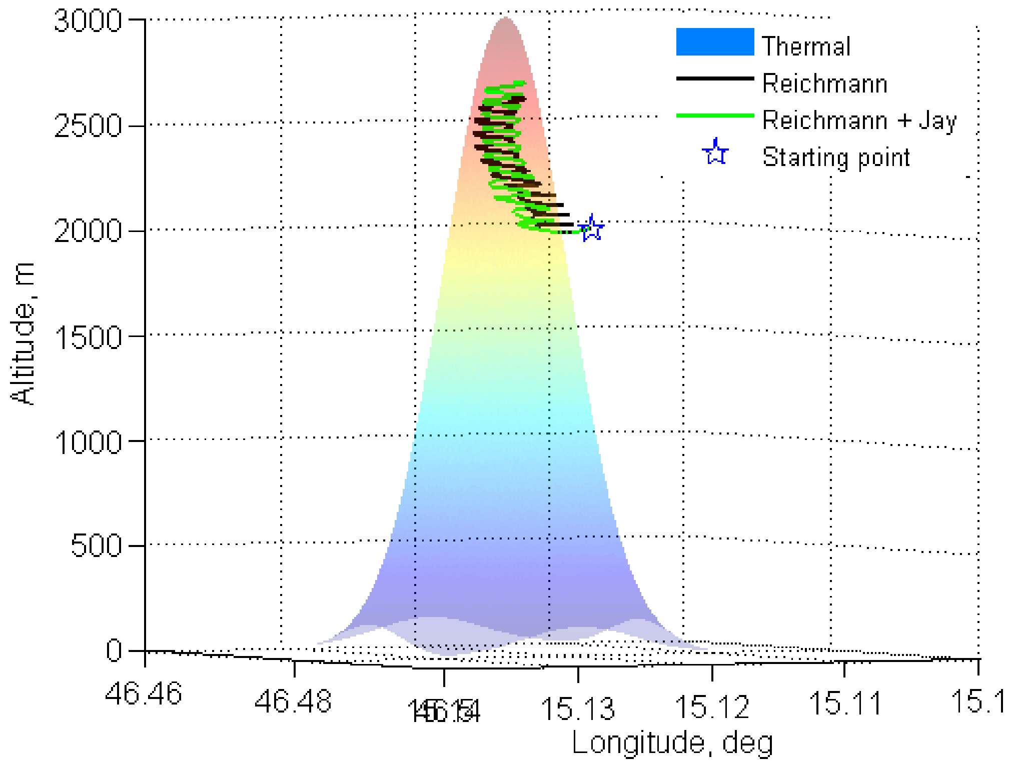


Figure 13. Paths of both gliders for performance comparison of controllers.

More details about the climb performance are presented in Figure 14; it shows the altitude profile versus the simulation time. Besides comparing the initial and final positions of the gliders in Figure 12, it is also possible to observe that both controllers generate a different slope of the climbing profile. The difference in altitude after 230 s of flight is about 90 m; therefore, if the gliders visit multiple thermals during the same period of time, the energy gained by each controller will be very different. Another important fact to mention is that thermals have a limited lifetime (they eventually disappear); therefore, it is better to climb as fast as possible when the thermal is detected.

Notice that these results were obtained using the simulated environment of Condor; therefore, it is still necessary to validate and verify the results in real flight tests because the geometry of a real thermal might affect the performance of both controllers.

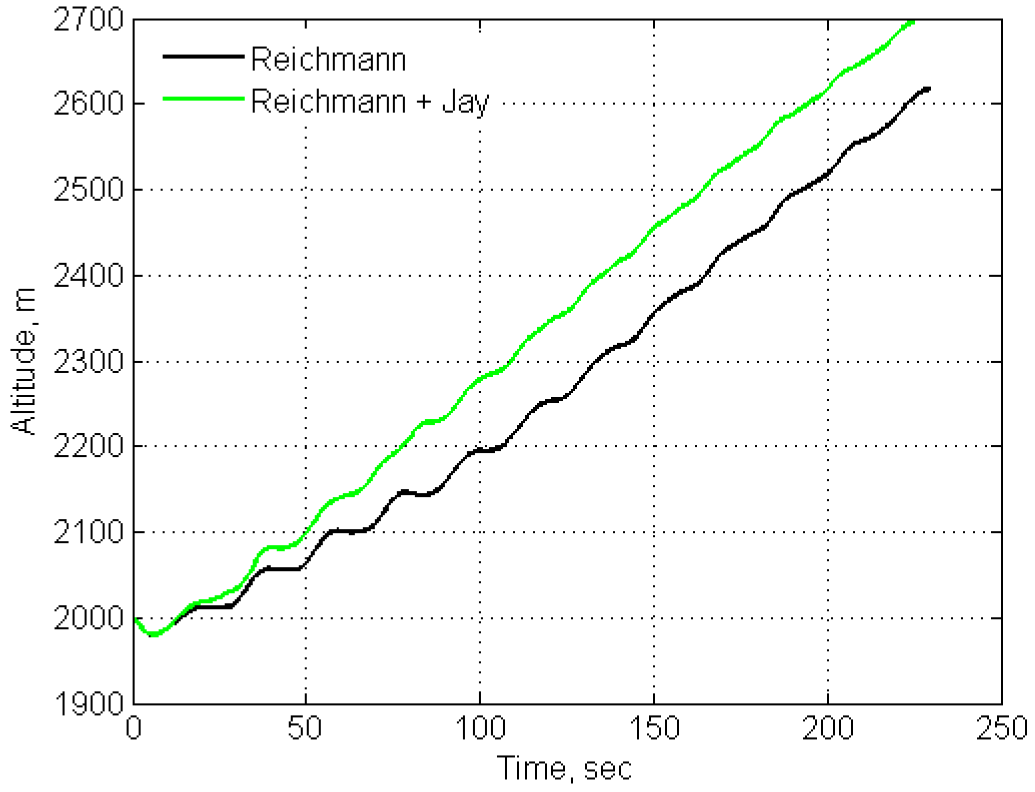


Figure 14. Evolution of altitude profile of two controllers.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. PATH PLANNING

The chapter proposes a Bayesian search approach as a solution of the problem of finding the best navigation path toward the thermals in the area of interest; this algorithm allows native integration of any prior knowledge about the atmospheric or geographical conditions that affect the creation of thermals. The contribution of this chapter is to minimize the energy spent in a search for thermals, which in turn maximizes the energy available for practical mission objectives. The novel navigation algorithm is designed and implemented in a flock of gliders. The evaluation of the expected performance of the resulting decentralized control architecture is performed in numerical simulation using Condor software. The results show that the distributed architecture that adapts the Bayesian search approach to finding the best path through the sequence of thermals is very efficient as it increases the endurance of multiple gliders.

A. OVERVIEW

The ability of a glider to detect thermals and gain potential energy by utilizing the thermal centering control still leaves unsolved the task of finding thermals. There are a number of techniques that guarantee completeness of search for a known feature in a given bounded search space. In application to the search for thermals by soaring gliders, the choice of the technique should account for the fact that when “not in thermalling flight,” the glider continuously loses altitude (potential energy). Since the objective of the soaring system is to minimize or better avoid using the battery powered propulsion (to preserve the battery energy for the overnight flight), the use of electric propulsion must be penalized. However, the time allowed for the glider to find at least one thermal might be too long, which will result in premature flight termination. The solution of the search problem adopted in this work is mostly biologically inspired—thousands years of evolution enable many species of birds to easily find thermals in the environment by integrating a seemingly disconnected set of features that birds observe in the environment. The engineering implementation of the biological inspiration relies on the Bayesian search technique. The Bayesian inference is a natural and a mathematically rigorous approach that allows integrating heterogeneous data and prior knowledge represented by probabilistic or statistical means.

Bayesian inference has been studied extensively by numerous researchers in the operations research area. This approach allows treating the search area as a set of discretized cells with a probability of finding a target associated with each cell—the probability map. Initial “assignment” of the probability values is based on the level of information available about the existence and the physical nature of the phenomena in each cell. If no information is available, then the prior probability is uniform and small; the integral of the probability over the area of all cells in the operational area is equal to 1. If there is a belief that some cells have higher probabilities of encountering the phenomena, then the corresponding probabilities are increased; still the corresponding integral over the entire area is 1.

In application to the search for thermals, the probability of finding a thermal is continuously evaluated by the glider which acts like a detection sensor. The dynamic update of the probability distribution map is based on the given prior probability and the measurements provided by the thermal detection algorithms. Thus, this chapter adapts the Bayesian inference approach and applies it to the case of collaborative gliders operated in the field of multiple thermals. The developed solution implements methods that initialize the prior probability distribution map, update the values of probability of each cell of the discretized map, and use the map to optimally solve the navigation task. The optimal navigation solution utilizes the well-known “travelling salesman problem” (TSP) algorithm. The navigation solution implemented onboard of each glider allows guiding each glider of the flock through the field of thermals in a given operational area. Each glider implements onboard the same navigation algorithm. Simultaneous sampling of the operational environment and exchange of knowledge (probability of the thermal in a sampled cell) over the communication network makes the overall solution highly distributed, which is a very desirable property of the cooperative thermal soaring system.

The practical objective of this chapter is to develop and implement a search algorithm that enables cooperative gliders to find and navigate through the field of thermals while also performing a specific mission in a given area. The software simulation results are used for the verification of the algorithms and evaluation of their efficiency and computational feasibility. Their analysis shows high efficiency of the Bayesian search that significantly reduces the number of visited points. The number of points and the cumulative distance travelled between them can be directly translated to the minimization of energy loss. It is also demonstrated that the distributed nature of solution keeps the probability maps onboard the gliders nearly synchronized; the higher the update frequency of the information exchange among the gliders, the smaller are the differences of the onboard maps among all gliders.

B. METHODS

This section explains the atmospheric phenomena as well as the characteristics of the terrain that contribute to the formation of thermals; furthermore, this information is used to solve the task of finding thermals over an area of interest.

1. Physical Nature of Atmospheric Thermals and their Mathematical Model.

The formation of thermals results from the interaction of the atmosphere with the solar radiation. The Federal Aviation Administration (FAA) in the *Glider Flying Handbook* (GFH) [34] provides a clear explanation of how thermals are formed and evolve over time with the changes in the atmospheric conditions.

There is a mixture of gases present in the atmosphere, with nitrogen and oxygen contributing up to 99% of these gases. Another important component of the atmosphere is the water vapor that varies its concentration depending on the location over the Earth. Besides the gases content and the water vapor, the current state of the weather is also determined by the vertical and horizontal gradients of the temperature, density and pressure; all of these parameters evolve over time thus producing the dynamics in weather conditions.

The atmosphere is divided into five layers: troposphere, stratosphere, mesosphere, thermosphere, and exosphere. The focus of the thesis is on the troposphere where most of the Earth’s weather is formed. The troposphere contains most of the water vapor; the lower part of the troposphere interacts with the land and sea surface that generate most of the thermals. An important property of the atmosphere is that its temperature decreases with the increase of altitude. The troposphere extends from the Earth’s surface to about 36,000 feet over the mean sea level (MSL). For our purposes, it is sufficient to assume that a prototype glider will not leave troposphere layer. Thermals are enclosed in the category of microscale atmospheric phenomena lasting from seconds to minutes; the best updrafts for thermal soaring are located in large heated areas, and their typical diameter varies from 500–1,000 feet (152.4 to 304.8 m).

Formally, a thermal [35] is a column of rising air produced by the convective effects which allow the atmosphere to transfer energy from a zone of high temperature (low altitude) to a lower temperature (high altitude) region. Therefore, thermals [35] are created by gradients of temperature between the earth’s surface and the surrounding air. Sunrays heat the earth’s surface unevenly due to the variation of angle of incidence, the reflection index of the earth surface. As a result, the air nearest to the hot spots gets warmer, thus decreasing its density and allowing it to become lighter. The convective effect (Figure 15) causes the lighter air to rise. During the upward travel the “hot bubble” is getting cooler through the thermodynamic interaction with the surrounding atmosphere; this ascending motion stops when the rising air reaches the same density (temperature) layers. Thus, the ascending motion of the thermal updraft is always complemented by the downwards motion of the air that effectively absorbs the energy of the rising.

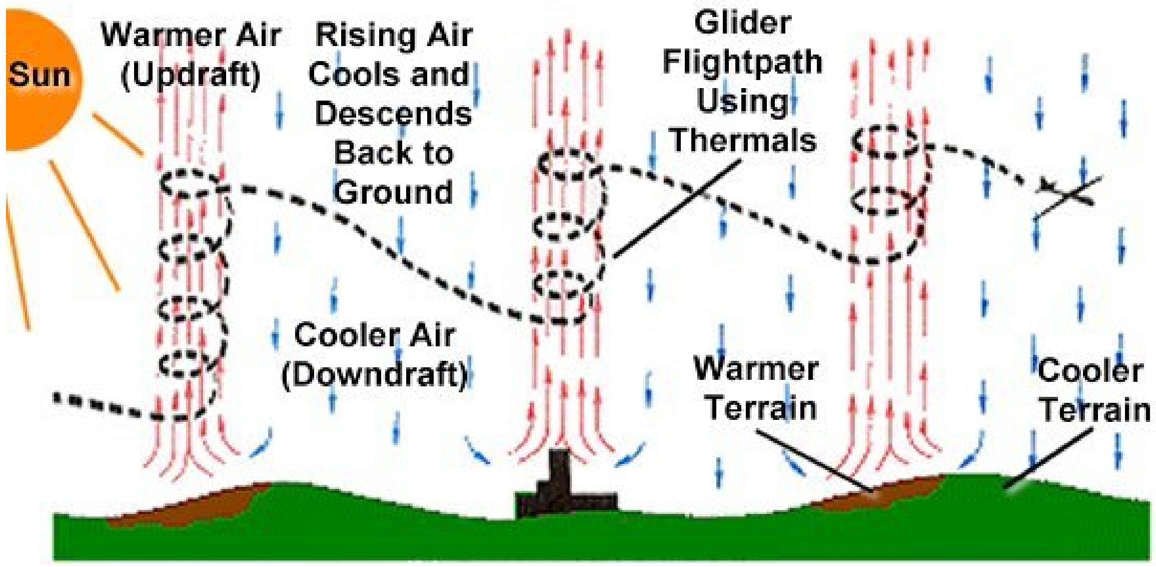

Sun
Warmer Air (Updraft)
Rising Air Cools and Descends Back to Ground
Glider Flightpath Using Thermals
Cooler Air (Downdraft)
Warmer Terrain
Cooler Terrain

Figure 15. Convective effect of thermals in the atmosphere, from [36].

Among numerous models of thermal updrafts, there are two particularly useful approaches (Figure 16)—bubble model and column or plume model. The type of thermal that is formed—bubble or plume—depends on the size of the heated surface and the temperature of the surrounding air near that surface; when the area is small, a single bubble is expected to form; while if the area is big, a plume might be generated. These two models encompass most of the characteristics and dynamic behaviors of thermals which should be considered by autonomous soaring gliders. It is also worth noting that although the bubble and plume models allow for unique mathematical description of all possible thermals, real life thermals are always different in their shape, maximum strength (strength at the core of the thermal), ceiling (altitude where the mass of air stops rising), and the lifespan.

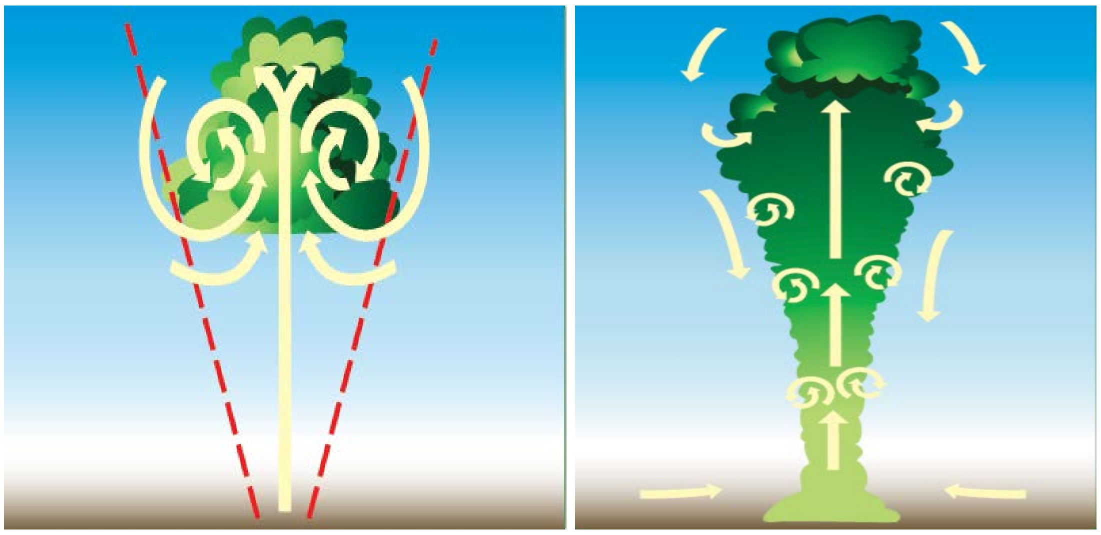


Figure 16. Bubble thermal model (left), and plume thermal model (right), from [34].

In both models, the air at the core ascends faster than the remaining outskirt mass. Figure 17 is taken from the *Glider Flying Handbook* [34]. It shows a cross section of a thermal that is common to both models; the darker green section represents the rapidly ascending core of the thermal and the red section shows the sinking zone at the outer layer of the thermal.

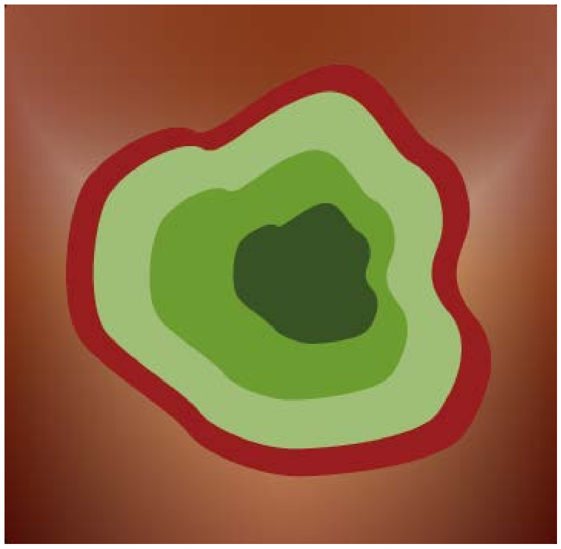


Figure 17. Cross section of a thermal with color-coded map of its strength, from [34].

In order to formally describe the characteristics and dynamic behavior of thermals, several authors have developed mathematical models that capture most significant characteristics of thermals. Allen [37] implemented a model based on experimental data sampled by rawinsonde balloons. This model adapts the parameterized Gaussian distribution equation to describe the bell shape of the thermal. The characteristics of thermals include the size, vertical velocity profile, spacing, and maximum height that are conveniently parameterized in two ways: the convective velocity scale w^*, which defines the upward speed of the thermal, and the convective mixing layer thickness z_i, which defines the maximum height-above-ground that updrafts can reach. Even though this model is effective in simulation of a single thermal and gives a sense of the energy that might be extracted from the thermal, it still does not account for the evolution of thermals over time, the influence of wind, and merging of thermals. Research performed by Zhenhua [38] addresses the time dependency and the drifting nature of thermals, thus providing significant improvement to the dynamic modeling of the thermal; see an example of the model in Figure 18.

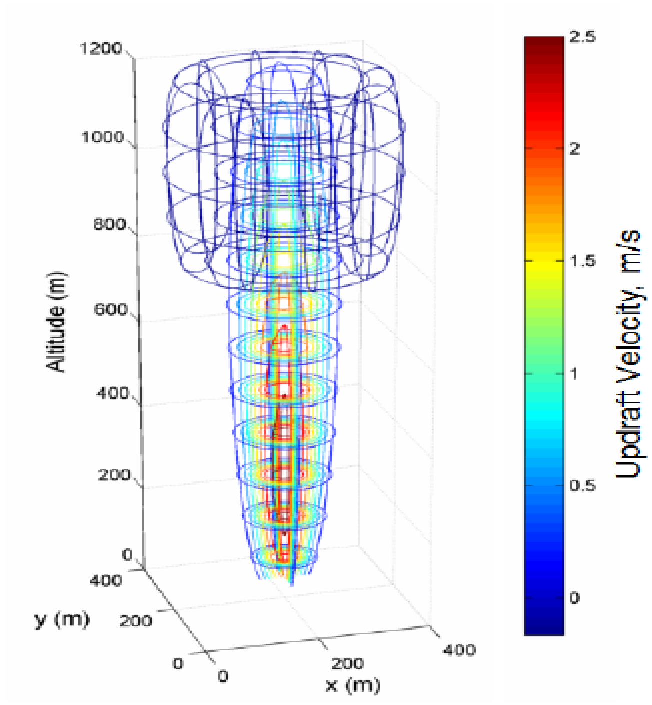


Figure 18. Model of a thermal implemented by Zhenhua, from [38].

Similar to the previous findings, the results of Cushman-Roisin [39] present a mathematical model of a thermal, which describes how its characteristics evolve over time. The model is the most advanced in presenting the dynamic updraft phenomena, it is therefore described by following the presentation in [39]. This parameterized model uses basic fluid dynamics principles; the dimensionless parameters are identified from the experimental data and allow predicting the evolution of a thermal over time. The main property of a thermal is its total buoyancy defined by Equation (4.1).

$$B = \alpha g T' V = g' V \tag{4.1}$$

where V is the volume of the thermal, T' is its temperature anomaly (the difference between the long-term average temperature and the temperature that is actually occurring), and $g' = \alpha g T'$ is the reduced gravity it experiences. This total buoyancy is a constant quantity; when the thermal rises, its temperature anomaly decreases by the dilution proportional to the volume increase. The volume of the thermal is expressed as in Equation (4.2).

$$V = mR^3 \tag{4.2}$$

where R is the radius of the thermal seen from above, and m is a volume coefficient that accounts for the slightly flattened shape, typically less than $4\pi/3 = 4.2$ (value for a spherical volume). This last term cannot be measured directly and is deduced based on some prior statistics. The mass conservation over time can be expressed as $\frac{dV}{dt} = Au$, where A is the enclosing surface area of the thermal and u is the average entrainment velocity across that surface. Taking the area A proportional to the square of thermal's radius R^2, and considering the entrainment velocity u to be proportional to the thermal's vertical velocity w, one can obtain Equation (4.3).

$$\frac{dV}{dt} = aR^2 w \tag{4.3}$$

where a is an experimentally defined dimensionless constant. Combining with Equation (4.2), (4.3) is reduced to Equation (4.4).

$$\frac{dR}{dt} = \frac{a}{3m} w \tag{4.4}$$

The momentum budget over time takes the form of Equation (4.5).

$$\begin{aligned} \frac{d}{dt}\left(\frac{3}{2}\rho_{thermal} Vw\right) &= Upward\_bouyancy\_force - Downward\_weight \\ &= \rho_{ambient} Vg - \rho_{thermal} Vg \\ &= \rho_{thermal} \alpha T' Vg = \rho_{thermal} g' V \end{aligned} \tag{4.5}$$

where the factor $3/2$ is due to the added-mass effect. Physically, the thermal also accelerates (time derivative of $\rho_{thermal} Vw$); thus it also accelerates the surrounding fluid that is diverted by its body. The bubble effectively accelerates up to 50% more fluid mass; hence, the factor $3/2 = 1.5$. Rearranging Equation (4.5), we obtain Equation (4.6).

$$\frac{d}{dt}(Vw) = \frac{2}{3} g' V \tag{4.6}$$

The term $g'V$ at the right-hand side of this equation is constant and equals to the total buoyancy, see Equation (4.1). Integrating this equation over time yields

$$Vw = \frac{2}{3} g'Vt = \frac{2}{3} Bt \tag{4.7}$$

where $t = 0$ denotes the time when the thermal had zero momentum. Next, solving for w ($w = 2Bt/3V = 2Bt/3mR^3$) and substituting into Equation (4.4) defines the dynamics of the thermal radius R ($dR/dt = \left(2a/9m^2\right)\left(Bt/R^3\right)$), which after integration results in the following:

$$R = \left(\frac{4a}{9m^2}\right)^{1/4} B^{1/4} t^{1/2} \tag{4.8}$$

Equation (4.8) defines the variation of the radius of the thermal over time and can be used to determine parameters of the thermal, such as volume V, vertical velocity w, and reduced gravity g' as expressed in Equations (4.9).

$$V = \left(\frac{64a^3}{729m^2}\right)^{1/4} B^{3/4} t^{3/2} \tag{4.9.1}$$

$$w = \left(\frac{9m^2}{4a^3}\right)^{1/4} \frac{B^{1/4}}{t^{1/2}} \tag{4.9.2}$$

$$g' = \left(\frac{729m^2}{64a^3}\right)^{1/4} \frac{B^{1/4}}{t^{3/2}} \tag{4.9.3}$$

The parameters R, V, w, z, and g' are used to describe the evolution of the thermal over time. The terms a and m contained in their equations are determined based on laboratory tests, such that their values only depend on B and t.

The singularity of equations (4.9) at $t = 0$ is purely numerical and does not represent any physical nature of the thermal formation, as it is assumed that the initial conditions of the thermal formation are: $V = 0$, $w = \infty$, and $g' = \infty$.It is also observed that Equations (4.9) depend on the dimensionless parameters a and m, which are experimentally determined by analyzing thermal properties. First property to be analyzed is the dynamics of expansion of the thermal radius with the vertical distance travelled, *z*. It can be obtained by integrating $dz/dt = w$ that yields Equation (4.10).

$$z = \left(\frac{36m^2}{a^3} \right)^{1/4} B^{1/4} t^{1/2} \tag{4.10}$$

Getting the ratio of the radius R to the elevation z we obtain the expression $R/z = a/3m$, which makes evident that thermals grow at similar rates. Utilizing the laboratory observations [39] presented in Figure 19, one can conclude that the ratio of R to z is about $\tan(14^\circ) = 0.25$; thus, $R = 0.25z$, and $a = 0.75m$. Another useful property to be observed is the ratio z^2/t (the time constant predicted by the theory), which varies from experiment to experiment in proportion to $\sqrt{B}$ (Figure 20). The theoretical coefficient of proportionality is $\sqrt{36m^2/a^3}$ and the experiments estimate its value at 5.80. Solving $a = 0.75m$ together with $\sqrt{36m^2/a^3} = 5.80$ yields: $a = 1.90$ and $m = 2.54$; therefore, the final equations that describe the characteristics of a given thermal are given in (4.11).

$$R = 0.60B^{1/4}t^{1/2} \qquad V = 0.55B^{3/4}t^{3/2} \qquad w = 1.20\frac{B^{1/4}}{t^{1/2}}$$

$$z = 2.41B^{1/4}t^{1/2} \qquad g' = 1.81\frac{B^{1/4}}{t^{3/2}} \tag{4.11}$$

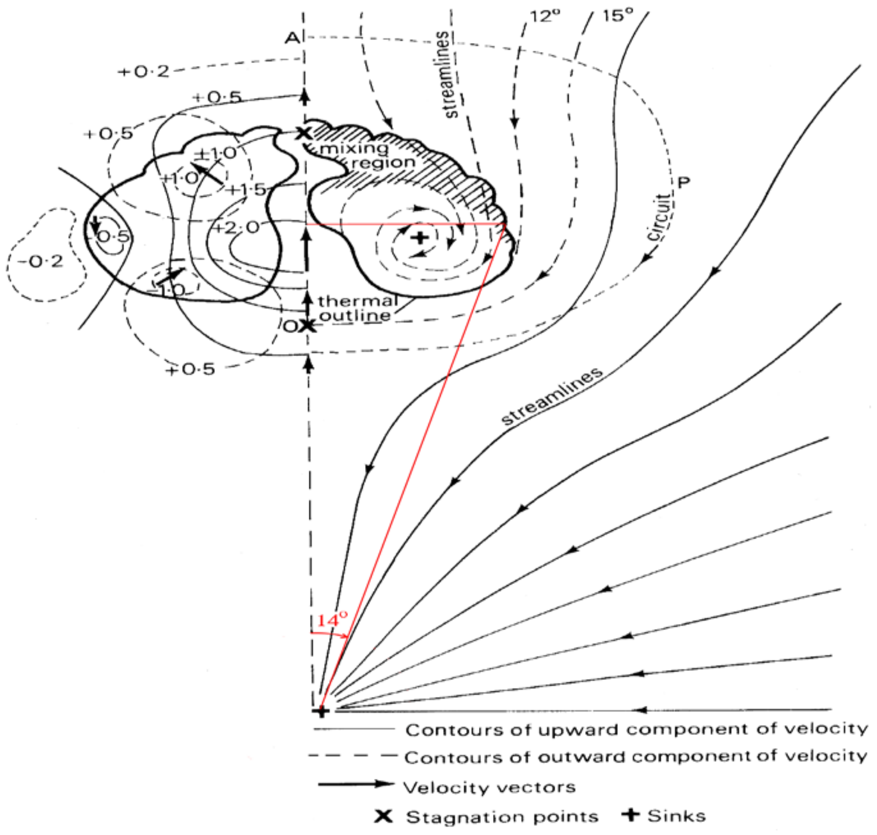


Figure 19. Anatomy of a rising thermal from [39]. Red line traces the outer edge of the thermal over time.

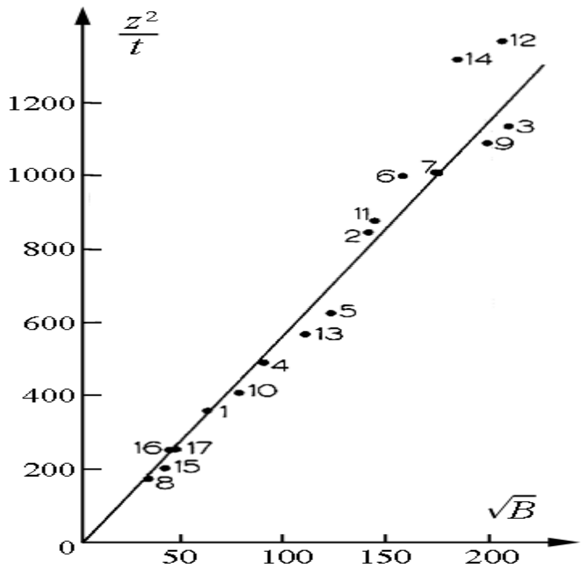


Figure 20. Plot of z^2/t (time constant during the life of the thermal) versus the square root of the thermal's buoyancy from [39].

To illustrate the expected results of the presented mathematical model, a numerical simulation of the evolution of two thermals over the period of 400 s with values of total buoyancy $B_1 = 15$ and $B_2 = 150$ is shown in Figure 21 and Figure 22.

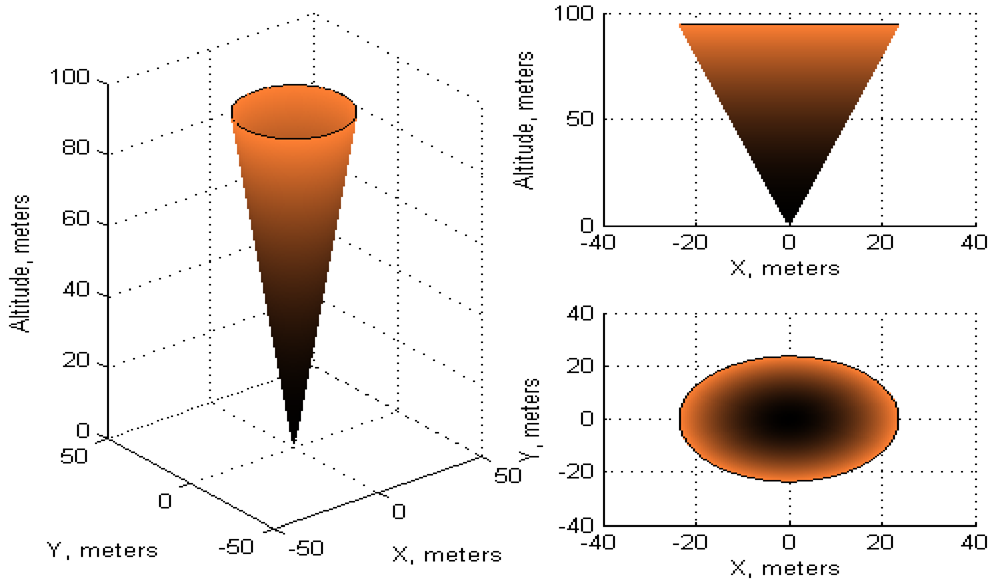


Figure 21. Evolution of a thermal's shape after 400 s using $B_1 = 15$.

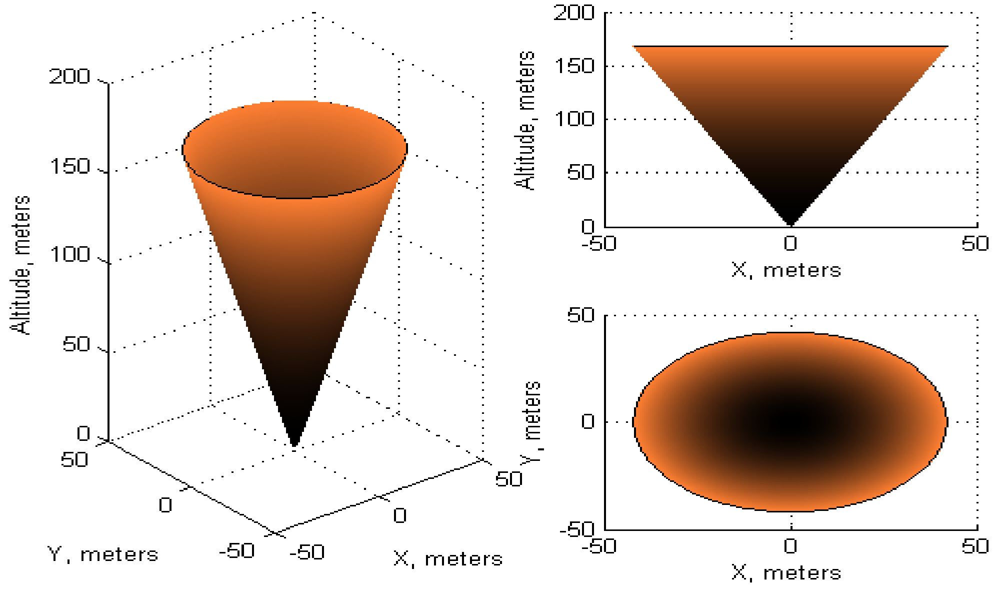


Figure 22. Evolution of a thermal's shape after 400 s using $B_1 = 150$.

2. Search for Thermals

After understanding the behavior of thermals and how they evolve over time, it is desired to search for them in an intelligent manner. As explained in the previous section, the formation of thermals depends on many factors. Intuitively, it is clear that the more knowledge (statistical observations) one has about the thermal formation and the more recent the knowledge is, the more accurate the estimation of possible location of thermals should be. The type of information that is effective in determining the more likely spots of thermals include: the elevation maps—data containing latitude, longitude, and elevation which inherently describes the characteristics of the terrain at the area of interest; heat maps—showing distribution of temperature over the surface of the earth at a given time; meteorological data—including measurements of atmospheric variables that lead to the formation of thermals; real-time infrared imagery of the earth's surface—a vision aid useful in determining the heated spots on the ground, which are a very good indication of the location of a thermal; and a database of previously observed thermals in the area.

A natural search method that can run in real time to enable finding a target and to utilize a number of prior sources of information is the Bayesian search; it uses the Bayes' inference recursive procedure to update the probability of a hypothesis given the real-time observations; in application to the thermal search the hypothesis is that a thermal is found. The knowledge used to update the probability is given by a number of sensors which detect the presence of a thermal. In the current project the detection mechanism is based on the Sink Rate Polar and/or the Total Energy techniques.

Implementing the inference takes a number of steps. The first step is to identify the area of interest and discretize it into cells; commonly, the size of the cells is determined by the sensitivity and the coverage of the sensor (detection mechanism) because it is not desired to leave any uncovered spots is a given area. However, the smaller the size of the cells (fine grid), the more computationally expensive the system becomes, as the number of cells to be iteratively updated will necessarily increase. Moreover, running the fine grid algorithms onboard will require not only excessive memory, but also more significant communication expenses to synchronize the knowledge preserved in the fine grid across the flock of gliders. Communication is one of the loads of the electrical management system that will be discussed in the following chapters.

The approach we use to select the size of the cell is based on the average lateral dimension of a thermal. Besides the memory and the synchronization expenses, the reasoning also accounts for the robustness of the estimation algorithms running onboard. If the thermal is sampled by a glider which is of much smaller size, then flying in the proximity of the thermal will give the onboard estimation and detection algorithms sufficient time to produce a reliable result. As a result, the Bayesian search algorithm will be more reliable and computationally efficient, which is an important performance metric of distributed systems.

After the discretization of the area is done, the next step is to implement the inference rule and to update a prior probability value of finding a thermal in every cell. At each update step a background computation process normalizes the cumulative probability over the area of operation at 1.

Bayesian search takes into account that the sensors are not perfect and can produce erroneous indication of updrafts. The update mechanism considers two parameters that describe the characteristics and performance of a detecting mechanism; we call it a sensor. The first parameter is the false alarm rate “α”, which is the number corresponding to the ratio of the instances when the sensor determines the thermal is present when it is not; on the other hand, the second parameter is the misdetection rate “β” which defines the ratio of the times when the sensor recognizes that the thermal is absent when it is present.

To illustrate the process of choosing the α – β rates, consider the total energy approach that detects the presence of a thermal. The output of the $\dot{E}$ estimation algorithm is a continuous signal, which, in the simplest case, can be conveniently converted into a binary signal (1 – there is a thermal, 0 – there is no thermal) by choosing a threshold of $\dot{E}$. In a nominal flight with no influence of convective air, the value of $\dot{E}$ is negative; it increases and becomes positive only when the glider enters an updraft. Therefore an intuitive threshold for detecting a thermal is $\dot{E} > 0$. In fact, determining the proper value of the threshold is the tuning knob for the flight operator. First, consider the case of weak thermals with the strength well below the threshold $\dot{E} > 0$. The value of *β* will be high because of significant number of misdetections. On the other hand, if the threshold of $\dot{E}$ is decreased even more, the high number of false alarms *α* will correspond to the detection of numerous weak thermals with insufficient "lifting" potential. Based on this intuitive analysis, the threshold for the glider will be assigned at $\dot{E} > 0$. Considering that the detection and estimation algorithms have enough time for convergence (the glider flies inside a cell with the size of a thermal), the values for *α* and *β* will be low; some experimental adjustment of *α* – *β* and $\dot{E}$ threshold can still be done when in flight.

Implementation of the Bayesian search is based on a simple algorithm that recursively updates the probability given the output of an updraft detection sensor, see Equations (4.12).

$$\begin{aligned} p\left(\text{thermal\_present}|\text{thermal\_detected}\right) &= \frac{(1-\beta)\,p_0}{(\alpha)(1-p_0)+(1-\beta)\,p_0} \\ p\left(\text{thermal\_present}|\text{thermal\_not\_detected}\right) &= \frac{(\beta)\,p_0}{(1-\alpha)(1-p_0)+(\beta)\,p_0} \end{aligned} \tag{4.12}$$

where p_0 is the prior probability assigned to the cell, and *α* and *β* are the known sensor parameters. When the probability in a cell is updated, the probability of all the other cells is affected by the normalization algorithm.

As stated in Equation (4.12), it is required to have a prior probability at each update step in order to compute the new one. At initial time $t = 0$ there should be an initial probability map assigned to capture the best previous knowledge about the area of operation; in the worst case scenario when there is no prior information available, each cell can be initialized by the inverse of the number of cells in the area. This initial probability is the guideline for the search task and must be as informative as possible. This thesis proposes a heuristic method to determine the best prior probability of thermals based on the characteristics of the terrain elevation. The motivation for using the 3D elevation approach is based on the theory of thermals formation as a function of topography [34].

To solve this task of building a 3D elevation map the project utilizes a mapping service of Google, Inc. The online service provides a mechanism for retrieving of elevation data at any point in the world. The company provides the Google Elevation API (Application Programming Interface) [40], which is a simple interface that allows a user to get elevation data over an area defined by geographical coordinates. Using advanced MATLAB capabilities, it is possible to automate the process of retrieving and collecting the desired data. Appendix E contains the MATLAB script developed for retrieving data from the Google Elevation API. Figure 23 shows an example of the capabilities of this implementation. Here, the figure represents the elevation map of a portion of Camp Roberts in California.

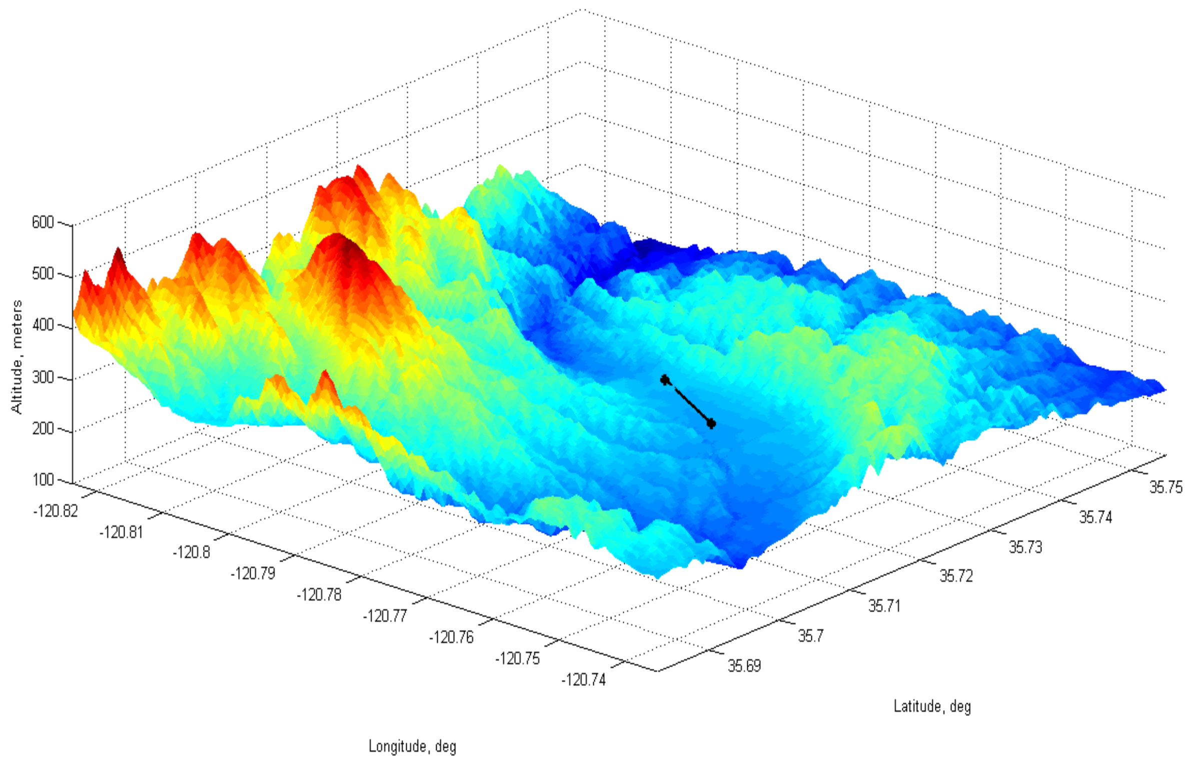


Figure 23. Elevation map of Camp Roberts, California.

The data of the terrain is also used for the discretization and analysis of the area including the slope characteristics of the terrain. According to the *Glider Flying Handbook* [34], thermals are most likely to occur over flat and hilly terrain; the cue is to look for sun-facing slopes. Unless the sun is directly overhead, the heating of a sun-facing slope is more intense than over the adjacent flat terrain (Figure 24) because the solar radiation strikes the slopes at higher angles. Also, cooler air usually stays in low-lying areas over night; then it takes longer to warm during the day.

Infrared imagery can also be used to identify the differential temperature distribution that is directly related to the likelihood of forming convective thermals. Black asphalt parking lots or roads can also produce strong thermals and usually they are located in flat lands as observed in Figure 25, where it is evident that there is significant difference in temperature between the asphalt and the surroundings. Another cue is that slopes tend to be drier than surrounding lowlands, hence they heats better. Integrating the observations about the formation of convective updrafts results in better quality probability maps and saves energy when gliders are in search-for-thermals mode.

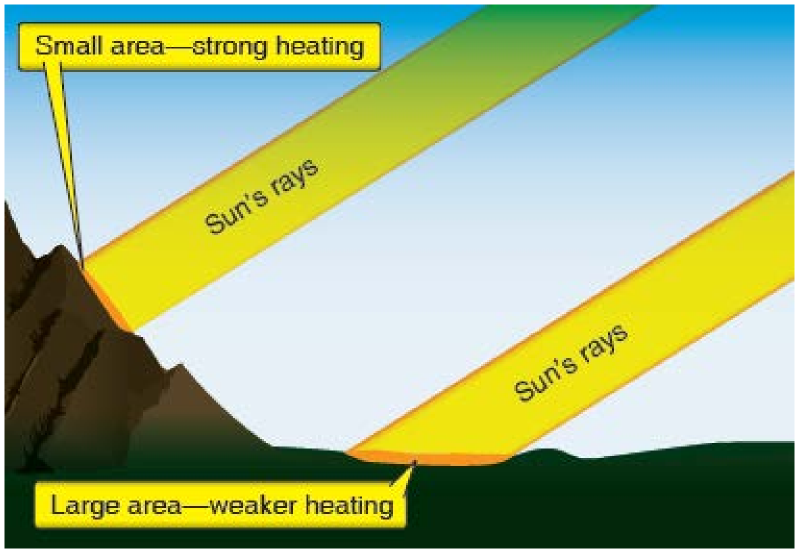


Figure 24. Effect of the solar rays on the terrain, from [34].

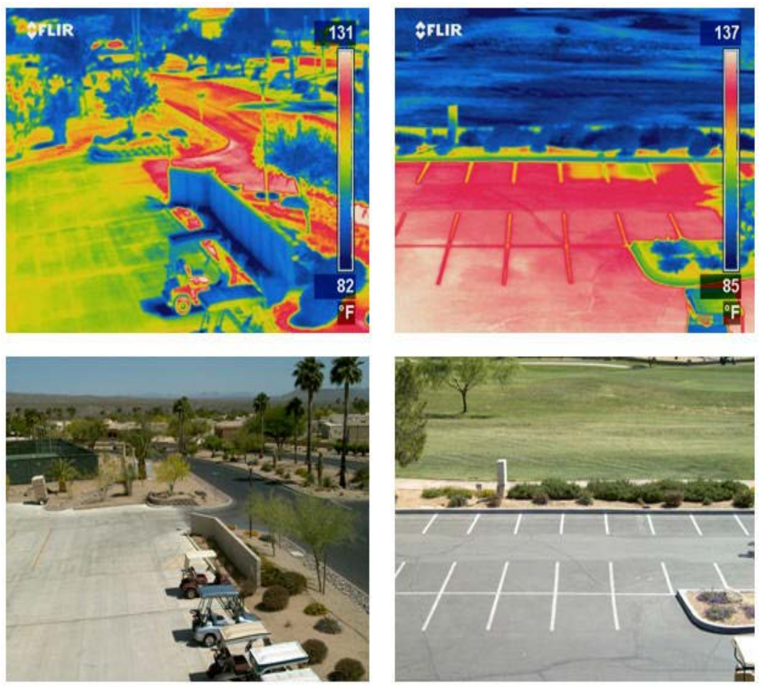


Figure 25. Temperature of the asphalt commonly located in flat lands, from [41].

Since thermals are likely to be formed on hills or flat land, finding the cells that meet this criterion can be done by analyzing the elevation map. Every point in the map is analyzed and three measurements are obtained: elevation e, slope e', and change in slope e''; the derivatives are defined with respect to the latitude and longitude. For the purpose of this research, the classification of cells with respect to the likelihood of generated thermals is based on the following inequalities: $e' < 3^\circ$ (flat lands), $e' > 30^\circ$ (hills), and $e'' > 25^\circ$ which is equivalent to the bottom of hills.

The discretized cells that satisfy these criteria are separated from the rest of the cells and assigned an equal probability of finding a thermal at that location equal to $p_{thermal\_high} = \left(0.9 / \text{number of cells that passed the criteria}\right)$, while the remaining 0.1 of the probability is distributed equally among the cells that did not passed the criteria $p_{thermal\_low} = \left(0.1 / \text{number of cells that did not pass the criteria}\right)$.

There are two equations in (4.12) given that there are two scenarios for generating a new probability based on Bayesian inference; as we are always looking for a thermal to be present, these scenarios are determined when the thermal is either detected or not detected. The probability $p\left(\text{thermal\_present}|\text{thermal\_detected}\right)$ is used only when the binary sensor determines that a thermal might be present, then the probability increases in the current cell; on the other hand, the probability $p\left(\text{thermal\_present}|\text{thermal\_not\_detected}\right)$ is applied when no thermal are detected, hence decreasing the probability in the current cell.

At each time step of operation the probability map of thermals is used to calculate the “shortest” path through the field of thermals in a given area of operation; at the initial time *t=0,* the system is assumed to be initialized with the prior probability distribution over the area of operation. The calculation of this path solves the navigation task of each soaring glider. Since the gliders are assigned a specific mission, the navigation solution should also account for the mission objectives. Mission objectives are represented by discretizing continuous areas that needs to be surveyed by the gliders of the flock. To accommodate a wide variety of possible ISR missions their objectives are represented by spatial and temporal primitives. Spatial primitives are the points or areas of interest (POI), while the temporal ones are the time requirements associated with each area; for example, an ideal persistent surveillance of a POI would require a rotation of gliders such that at least one of the gliders is at this POI continuously. Finally, the ultimate navigation solution is the one that calculates the energy efficient path for the mission objectives and optimal path through the field of thermals. This task is solved by adapting and implementing onboard of each glider a TSP algorithm, which provides a suboptimal solution to the desired navigation task.

Finally, the cooperative flock of multiple gliders executes a number of identical algorithms that allow “sensing” of the environment and executing the mission objectives. System identification (sink rate polar), updraft detection (sink and energy based), thermalling guidance, Bayesian search, TSP-based navigation, knowledge sharing, and task allocation are all examples of the algorithms implemented identically onboard of each glider regardless of its mission specific configuration (IO/EO sensors, for example). Thus, the distributed nature of the implemented solution improves robustness of the entire flock to failures. Therefore, the possibility of loss of a single agent will not result in losing the entire capability of the flock, but will result in its graceful degradation. This last feature is not only a desired property of any system of systems, but also a good metric that can be used at the mission planning stage.

3. SIL Environment – Implementation Details

The algorithm using Bayesian search is implemented over an operational area of approximately 12 x 12 km. The size of the cell 300 x 300 m is chosen to fit a generalized thermal updraft; the size is based on the statistical observation of thermals in California. The algorithms of Bayesian search are implemented in MATLAB scripting language; see details in Appendix E. The likelihood of finding a thermal in a cell is based on the estimation of the elevation and its derivatives (e, e', and e''); see the corresponding discussion earlier. The synchronization of the probability distribution map, which is continuously updated by the Bayesian procedure as the gliders perform their mission, is based on indexing the cells rather than referencing it by the associated latitude and longitude of the center; indexing requires one single integer while geographic coordinates require two floating numbers. The choice minimizes the number of data (number of bytes) to be exchanged and therefore reduces the communication bandwidth that in real operation additionally saves electric energy for communication. The simulation is performed for two gliders that share information at the update rate of 1 and 10Hz; two rates were chosen to build a tool that evaluates the influence of mis-synchronization of knowledge and its impact on the overall performance. The collision avoidance of multiple gliders is solved in two phases. First, it is performed locally by each glider that implements the "rights of way" procedure typical for manned aviation [42] but scaled down to 50 m of spatial separation. Second, the initial assignment of areas of responsibility intentionally separates the gliders to minimize the occurrences of midair collision. Although the implementation might not be optimal, it is not the major objective of the current research.

C. RESULTS AND DISCUSSION

Figure 26 shows the elevation map of the selected area of operation; this elevation map implements the maximum resolution allowed by free service provided by the Google API; that is, 25,000 points per request. The resulting lateral separation between sample points is approximately 75 m. This spacing for the cell size of 300 x 300 m provides about 25 sample points for each cell; see an example of a color-coded (by elevation) operational map in Figure 27.

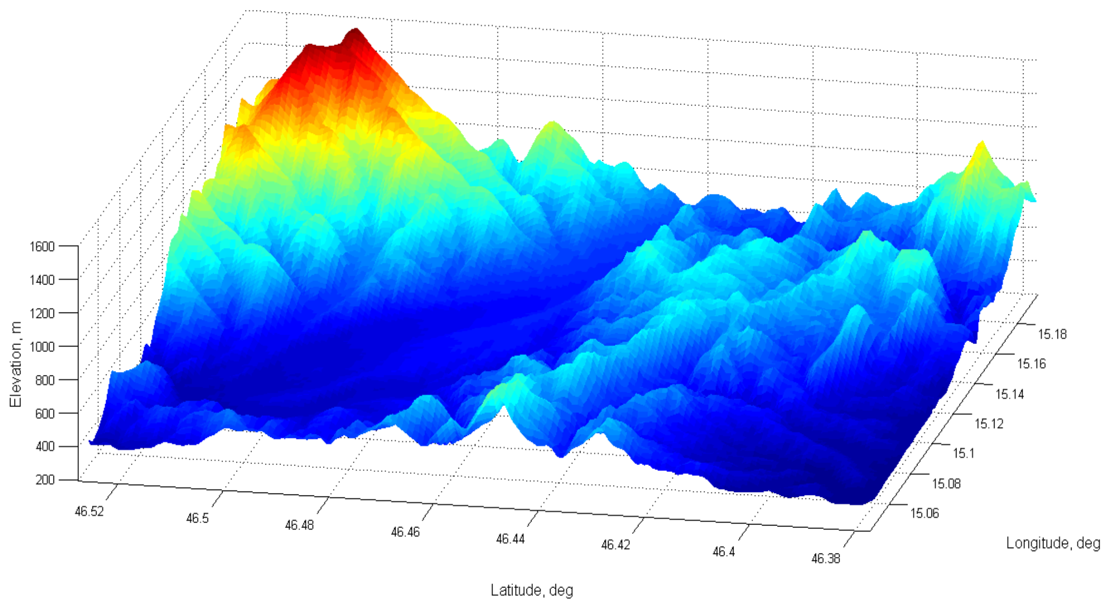


Figure 26. Elevation map obtained through the Google Elevation API.

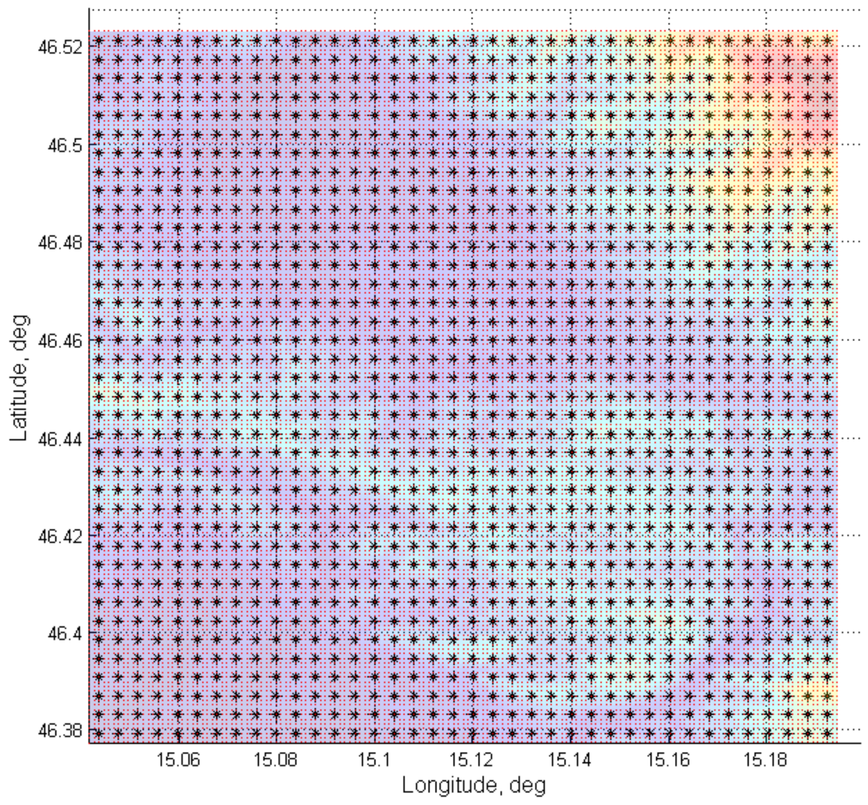


Figure 27. Discretized area in small cells (300 x 300 m) capable of enclosing a thermal.

The resulting 25 samples per cell provide smooth approximation of the slope corresponding to every sample. The average slope e' of the cells is presented in Figure 28; from this plot, it is easy to correlate with Figure 26 and match visually the type of topography given by its slope.

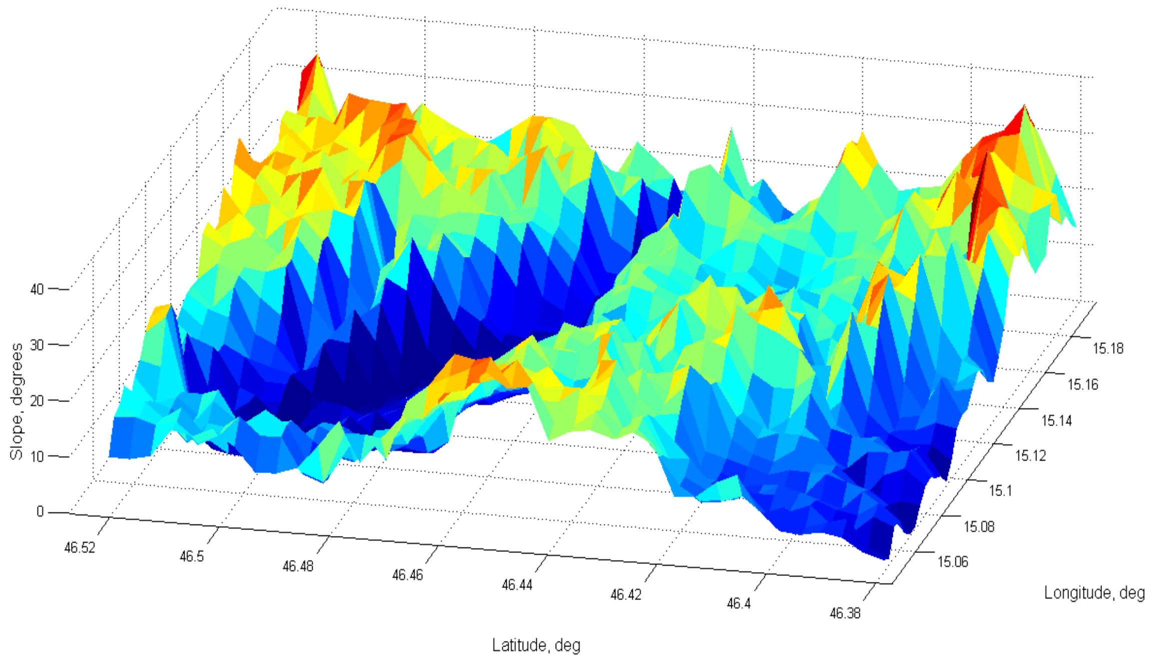
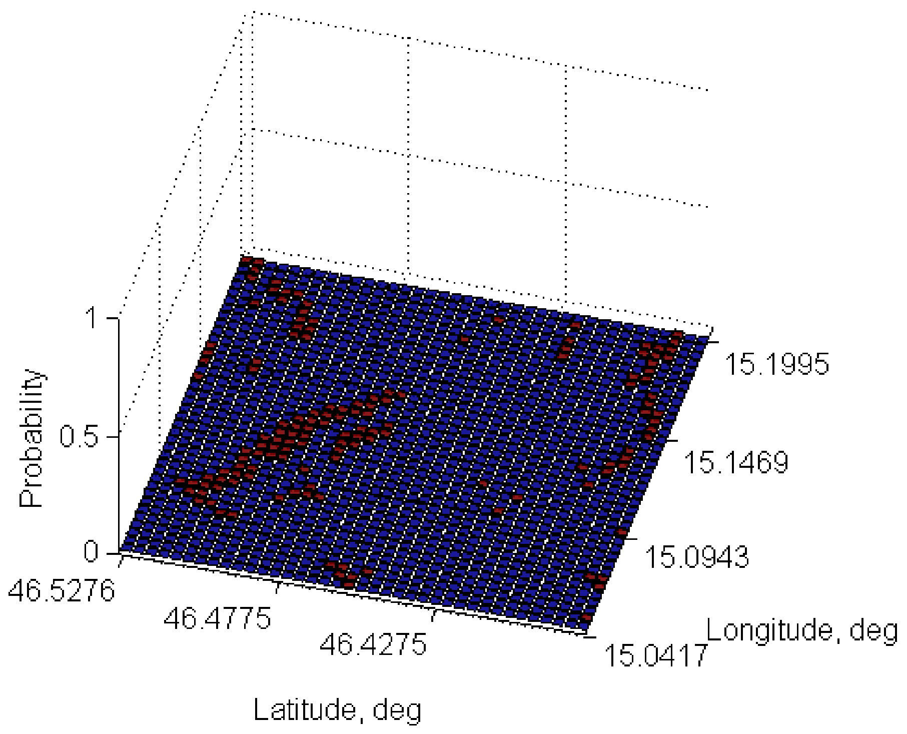


Figure 28. Average magnitude of slope per cell in the discretized area.

Calculating the slope e' of each cell and categorizing each as the “hills” or the “flat land” allows constructing the probability map of operational area. A figure of the color-coded prior probability map (it is used to initialize the mission planning task) is provided in Figure 29. The red cells represent the spots with the higher probability of finding a thermal; they are the most likely points to be included by the solution of this TSP problem.



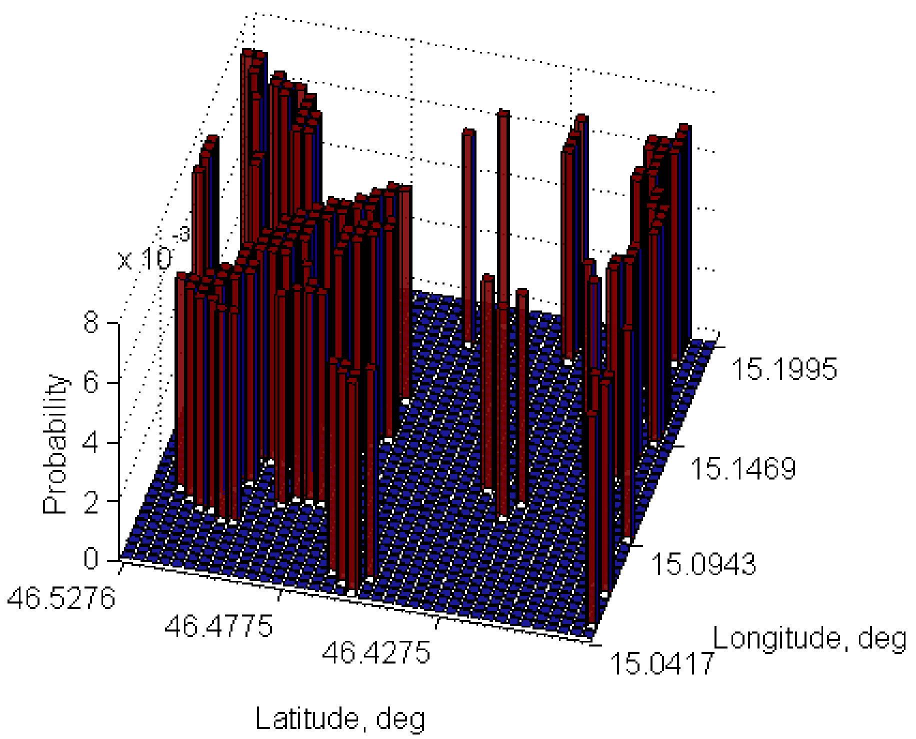


Figure 29. Probability map (top) with initial probability distribution given by the identification of “hills” and “flat land.” The bottom picture shows a zoomed-in region of the probability map.

The solution of the TSP task for every glider is given as a series of waypoints starting from the initial point of the mission; the point closest to the launching location of the glider. Figure 30 illustrates the paths of two gliders, which were computed independently onboard each airplane. Each glider covers a subset of the entire area to avoid visiting the same points; this also contributes to the collision avoidance.

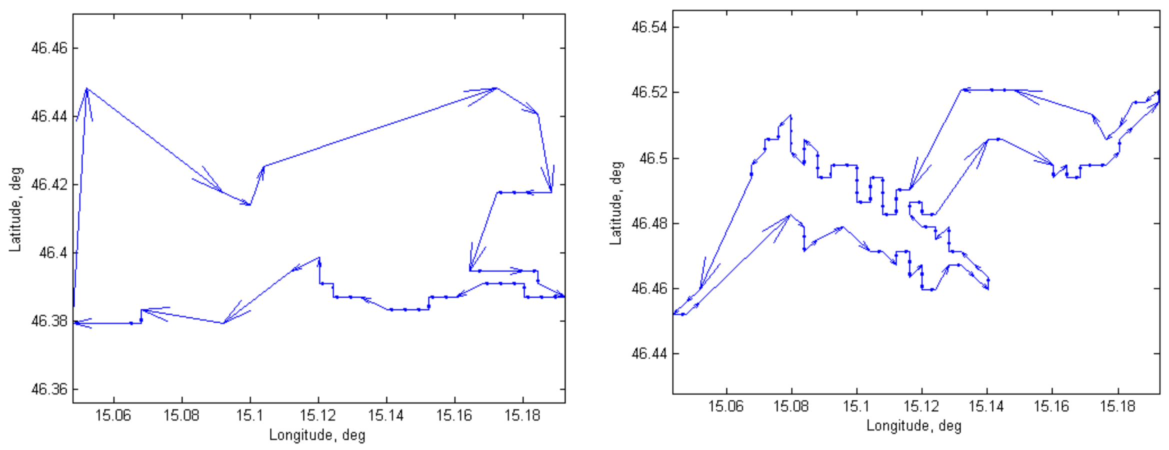


Figure 30. TSP solution for both gliders. Left picture, glider 1; right picture, glider 2.

In the first simulation, the objective is to (i) test the implementation of the Bayesian search in a decentralized architecture and (ii) observe the “similarity” of the probability maps when the knowledge synchronization rate is 1Hz. The results are presented in Figure 31; they explicitly represent the minimal discrepancies due to the update frequency. Furthermore, the analysis confirms that the knowledge based on the priory processed elevation map is useful, as several compact areas with a high concentration of thermals are discovered; comparison with the true data from Condor confirms the location of thermals. A very similar result is obtained at the update rate of 10Hz that suggests that the dynamics of thermals is significantly slower and the synchronization rate does not need to be enforced.

In the second simulation, the environmental conditions were changed to obtain a different pattern of the formation of thermals. The ability to change the local atmospheric conditions and to continue testing the accuracy of the Bayesian search along with the knowledge representation across multiple soaring gliders is an invaluable capability gained by integrating the control development environment with the high-fidelity simulation of Condor. The results of this simulation, shown in Figure 32; besides confirming the efficiency of the Bayesian search, they also illustrate minor improvement in synchronization of probability maps across the gliders. The discrepancy is minimally improved for the price of significantly increased communication bandwidth.

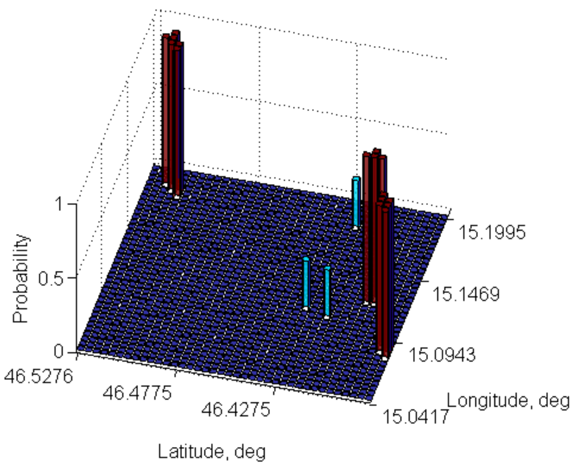


Figure 31. Probability maps for both gliders with a transmitting frequency of 1Hz. Left picture, glider 1; right picture, glider 2.

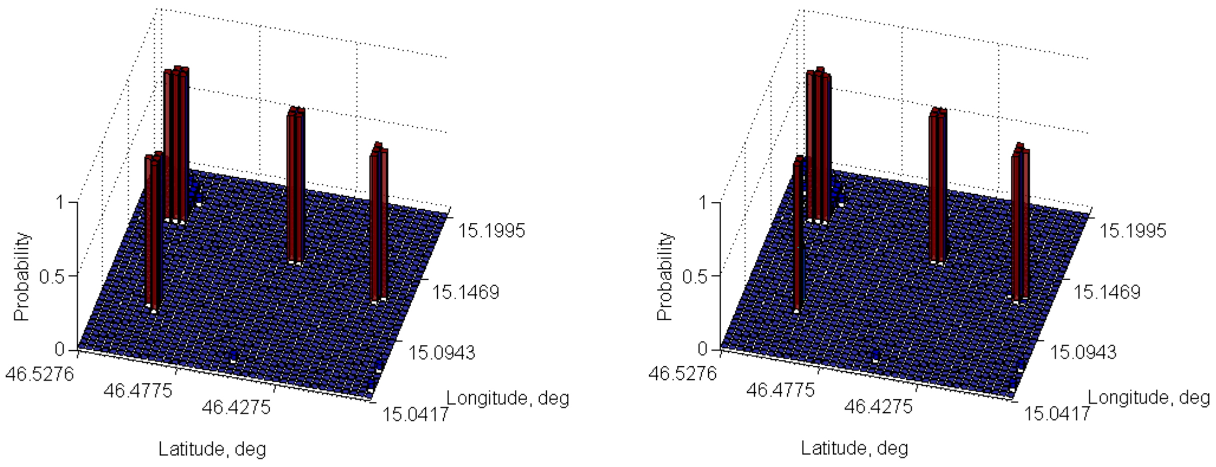


Figure 32. Probability maps for both gliders with a transmitting frequency of 10 Hz. Left picture glider 1, right picture glider 2.

THIS PAGE INTENTIONALLY LEFT BLANK

V. ONBOARD INTEGRATION OF SOLAR POWER

This chapter describes the design and the implementation steps of integrating the solar power onboard. The focus is on the detailed presentation of major components and their conceptual hardware and software integration onboard. While the energy extracted from thermals extends the endurance through the increase of potential energy, it is the photovoltaic energy that provides electrical power to flight-critical onboard instrumentation. An Electrical Energy Management System (EEMS) that is required to control the accumulation and distribution of the electrical energy is designed. The hardware prototype is built and tested in a multiday experiment. Several battery chemistries are tested to select the best performing configuration. The experimental results confirm the overall feasibility of the onboard EEMS architecture, the choice of components, and their optimal onboard configuration and placement.

A. OVERVIEW

A soaring glider has a number of onboard components that require electrical energy for safe flight, communication, and support of mission-specific sensors. The sole source of renewable electrical energy onboard of the proposed glider is the set of PV panels that are embedded into the skin of the wings.

Although there are examples of airplanes that exclusively use either PV energy or the convective air energy to sustain the flight, a flight proven technology which successfully combines both sources has not been reported yet. In 2011 Barnes et al. [20] proposed a project to harvest energy using photovoltaic, convective updraft, and the regenerative soaring sources (the motor of the UAV is used to generate electrical energy) when the airplane is in thermalling mode. Due to a number of obvious multidisciplinary challenges, the initially proposed effort has not yet produced any results implemented in flight. Some researchers have merged successfully different sources of energy. An example of this is the study carried out by Anton [19] who combined the vibrational energy from the airframe with the PV energy. The objective was to verify the feasibility of the implementation, and not to achieve a 24/7 flight.

As stated previously, the principal difference in the design of the NPS glider is in combining onboard two forms of solar energy (convective updrafts and photovoltaic) with the goal of performing autonomously a specific cooperative mission over an extended period of time. The practical task of integrating the photovoltaic and EEMS components onboard is solved in two steps. First, it is required to integrate the semi-rigid, mono crystalline, silicone cells into the wings at the stage of their manufacturing to minimize the adverse effects of parasitic drag. In the second step, the design and implementation of the EEMS that manages the electrical energy onboard need to be performed. The efficiency of the final design (fully-assembled glider) that implements the EEMS in soaring flight needs to be verified. The results of initial rigorous ground testing experiments are presented in this chapter.

B. METHODS

A conceptual diagram of the electrical system of a generic battery-powered UAV is given in Figure 33. This traditional concept assumes that the only source of energy for propulsion, avionics, and sensors is the set of onboard batteries that have finite energy capacity; therefore, the operational endurance of the aircraft is fundamentally limited. In order to eliminate the constraint of limited electrical power onboard, the proposed design integrates a capability of continuous recharging of onboard batteries during the daylight operation. The feasibility of the reasoning behind the concept is based on the fact that solar radiation is the sole source of both the convective air and the photovoltaic effects that coexist during the daytime. The concept assumes that during the nighttime, the electrical energy accumulated during the day will be used to sustain the overnight flight.

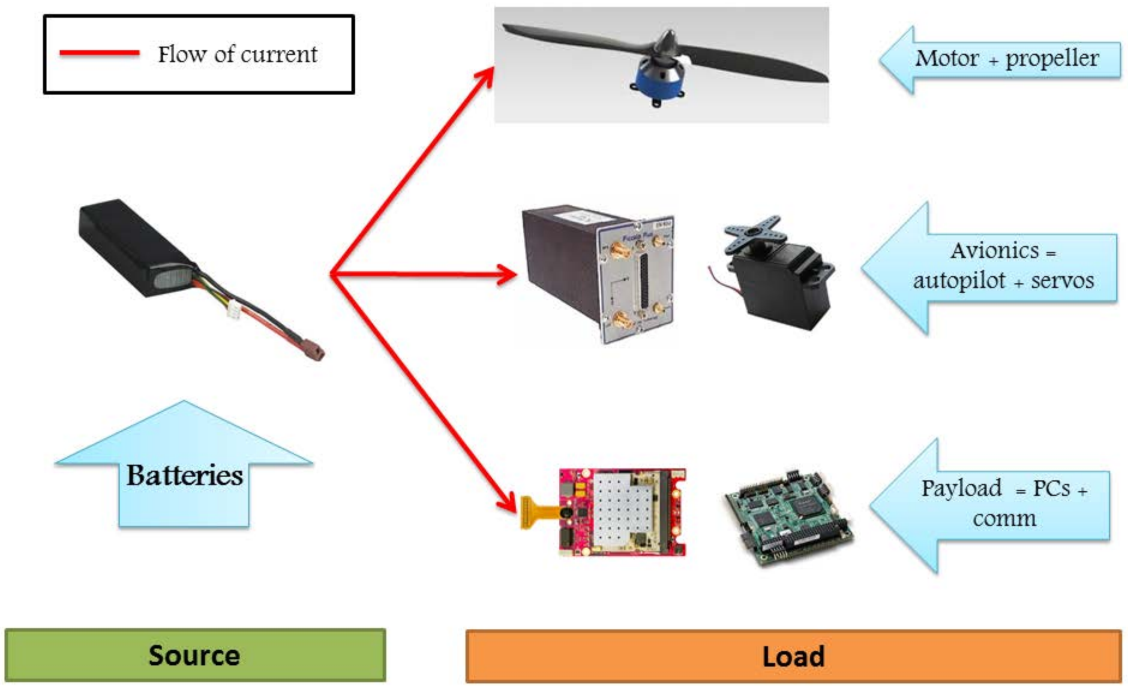


Figure 33. Conceptual block diagram of the conventional electric-powered UAV.

To enable the controllable harvesting, storage, and distribution of the electrical energy the EEMS is designed as a research and development platform. The primary objective of the EEMS prototype design is not only to provide the storage and distribution of electrical power, but also to enable access to key parameters of the architecture that characterize the system performance; the ability to perform case studies is the enabling capability of the NPS glider as a research platform. Figure 34 shows a block diagram of the EEMS architecture; the motivation behind this particular design and the selection of components is explained further in this chapter.

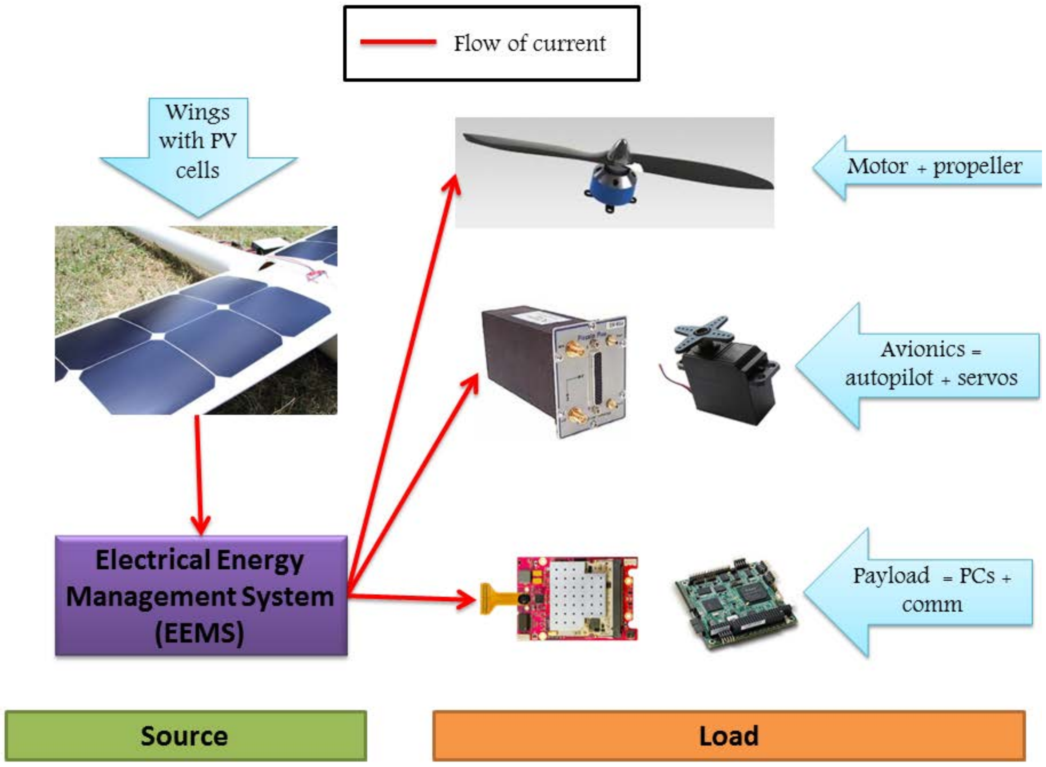


Figure 34. Block diagram of the EEMS system of soaring UAV.

1. Photovoltaic Cells

The photovoltaic or solar cells [43] are electrical devices that directly convert light to electricity at the atomic level; this phenomenon has been observed in a number of materials including crystalline silicon (high purity silicon), cadmium telluride, copper indium gallium selenide, and gallium arsenide. When exposed to sunlight, these materials exhibit the photoelectric effect, which is the absorption of photons of light with the release of free electrons. These free electrons produce an electric current. Figure 35 illustrates conceptually the physical process in the cross section of a solar cell material. Chemical composition and properties of the material are the key parameters that define the efficiency of the electrical energy generation.

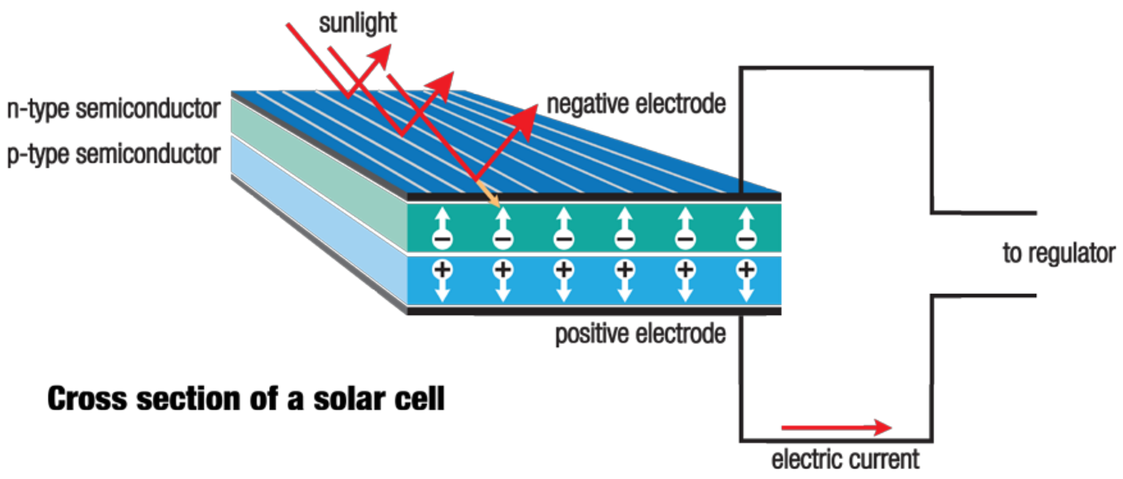


Figure 35. Cross section of a solar cell, from [44].

The NPS soaring glider utilizes a special type of solar cells; they are made of the pure mono-crystalline silicon [45] by utilizing thin film technology. At 22.5% efficiency and made flexible enough to be embedded into the glider's wings and sustain aerodynamic load in flight, they provide about 60 W of continuous power to be optimally exploited by onboard EEMS. The process of embedding the solar cells into the wings was not the objective of the thesis work; in fact, the manufacturing was done by an external provider. Nevertheless, some details of the construction process are described in Appendix B. In the current design, the solar wings contain 18 PV cells [45] connected in series with the following manufacturer specifications: efficiency = 22.5%, voltage at maximum power point = 10.476 V, current at maximum power point = 5.93 A, and maximum power = 62.122 W.

2. Electrical Energy Management System (EEMS)

This system controls the collection, storage, and distribution of the electrical energy onboard in an optimal and safe way. Besides achieving the utility of harvesting and storing the energy, it also manages a number of other tasks like:

- enforcing the extraction of maximum energy from the PV cells
- charging the onboard batteries
- protecting the batteries from an abnormal operation: over-charging and over-discharging; limiting current when charging or discharging

- balancing the battery cells during charge/discharge
- measuring the current and voltage at every branch of the electrical system to predict the amount of energy available for safe flight.

The laboratory prototype of the EEMS also has an extended set of capabilities that include:

- collecting and transmitting a set of measurements over Transmission Control Protocol/Internet Protocol (TCP/IP)
- determining the aging and state of charge of the batteries
- logging all the data processed by the EEMS

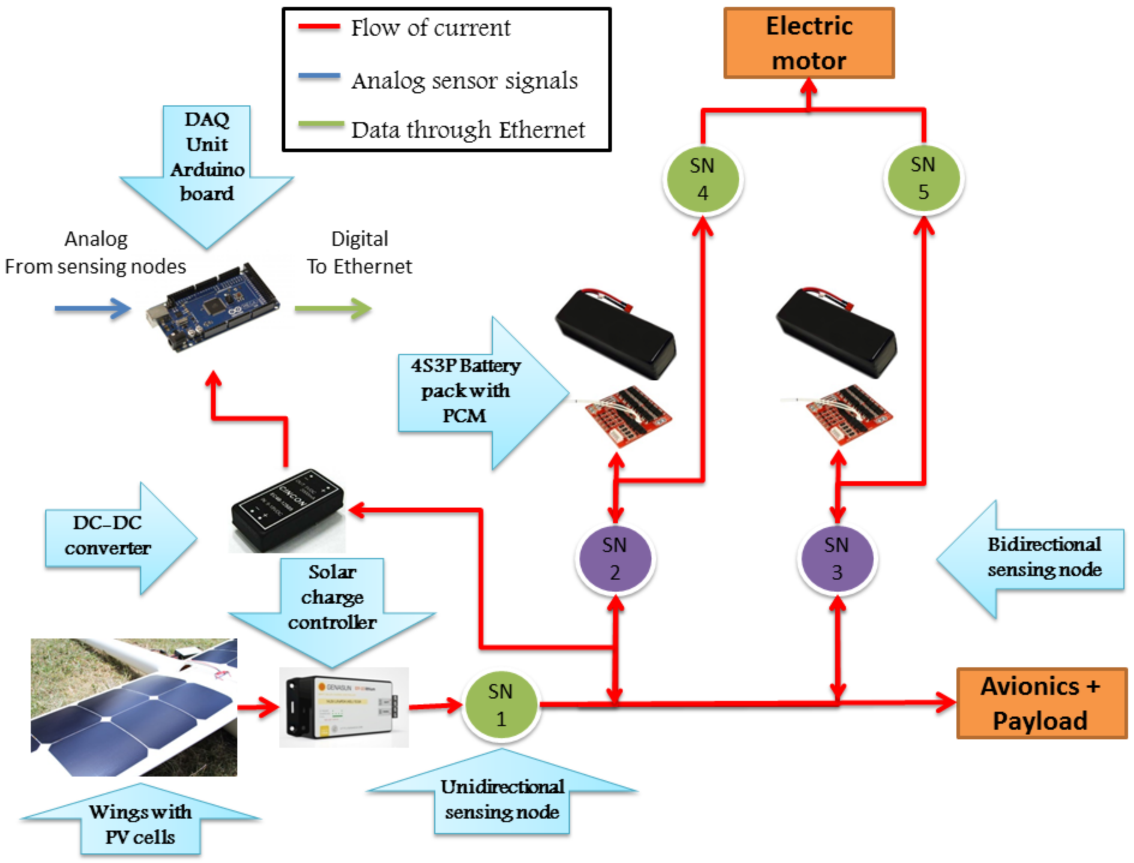


Figure 36. Block diagram of the EEMS.

The EEMS contains the integrated solar charge and the maximum power point tracking (MPPT) controller, which extracts the maximum available power (voltage and current) from the PV cells. The output of the MPPT unit is a constant voltage with variable current. The research and development prototype is equipped with five analog sensing nodes that measure voltage and current at different points of the architecture (see Figure 36). The unidirectional sensing node #1 measures the outputs of the MPPT controller in the primary “source” branch. Two other identical “storage” branches are used to connect power to the battery packs; each battery pack has its own Protection Circuit Module (PCM), which isolates the battery pack from the system in the case of off-nominal condition. There are two bidirectional multichannel sensing nodes #2 and #3 which measure the current/voltage in each “storage” branch. The “load” branch is used to power the avionics and the payload. The “storage” branch of each battery has a switching unit that is used to power the electric propulsion motor when directed by the onboard Guidance, Navigation and Control (GNC) algorithm. The voltage/current which enables propulsion is measured by the unidirectional sensing nodes #4 and #5. Finally, the “DAQ (data acquisition)” branch is used to power the Arduino central processing unit (CPU) board [46] which processes the analog measurements from all the sensors; this board also implements the estimation of the state of charge (SOC) of the batteries and predicts their aging. Furthermore, Arduino encodes all the onboard EEMS measurements and shares (transmits) them via User Datagram Protocol (UDP) with online monitoring procedures running either onboard or remotely on the ground.

Finally, it is worth noting that the EEMS configuration is designed to provide robustness to possible failures of onboard batteries. The EEMS provides power to the onboard avionics and propulsion motor (when necessary) in the presence of failure of one of the battery packs.

Solar Charge Controller with MPPT

The voltage and current outputs of the given configuration (18S) of solar cells directly depend on the amount of solar radiation received. This behavior is described by the family of curves shown in Figure 37, which were taken from the specifications of the chosen solar cells (SunPower, [45]). There is a point on each curve (marked by a circle) where the power is at maximum $P_{\max} = V \cdot I$; the point is located at the bending portion of each curve. The MPPT is an integrated circuit which automatically locates that point and tracks it to guarantee the maximum power output of the solar cells. The integrated solar charge controller, in turn, provides the regulated voltage required to charge the batteries and to power the various onboard loads.

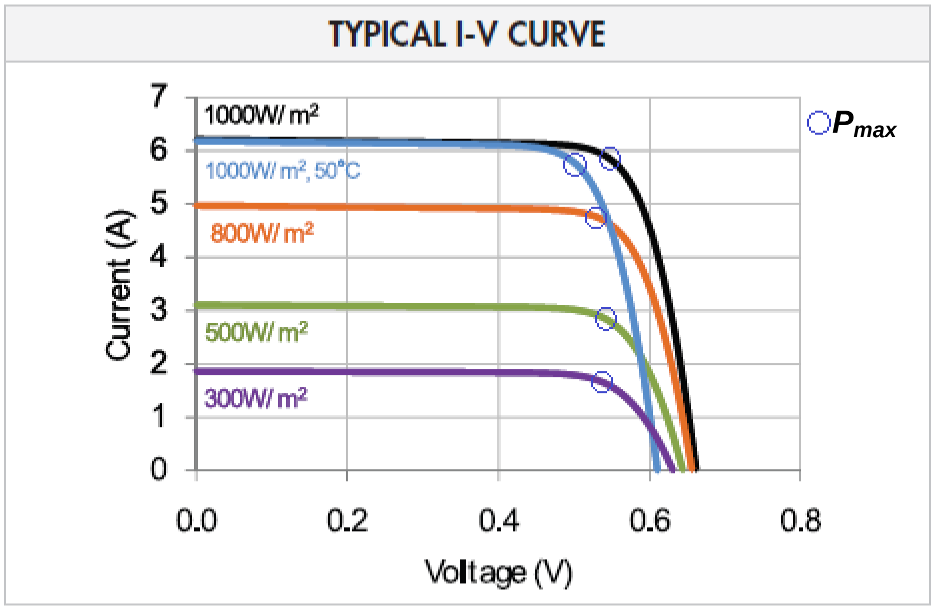


Figure 37. Voltage-Current curve for SunPower solar cells, after [45].

Battery Pack with PCM

In the current design there are two battery packs integrated onboard; these two packs should be capable of supporting up to six to eight hours of autonomous flight that is required for the overnight soaring. With about 60W of energy produced by 18 solar cells during the daytime, and only 18W required to power the onboard avionics (excluding the electric propulsion), the resulting excess of 42W will be used to recharge the onboard battery packs. These battery packs serve the following objectives:

- store excess energy coming from the solar cells

- provide energy to power the electric propulsion motor; in the actual design there is no need to use electric energy direct from the solar cells to enable the electric propulsion
- power the entire system during the night time

To accomplish these goals, the chemical composition of the battery packs must be selected such that they have the highest *specific energy* [47], $\text{specific energy} = \left(\text{energy stored}/\text{weight}\right)$, and are capable of managing high charging/discharging rates. A high discharge rate of up to 30 A is expected during the use of the electric motor. Also, to guarantee convenience, safety, and uniform handling procedures it is highly desired to utilize the same chemical composition of all battery packs. Furthermore, the same set of packs is used to provide electrical energy to all power loads onboard.

In order to choose the optimal chemical composition, a number of commercially available batteries were analyzed against the performance requirements. At the end, two chemical types were chosen and analyzed in detail: lithium-ion and lithium-polymer. The lithium-ion chemistry claims to provide the highest specific energy; however, its charging/discharging rates are very low; i.e., the amount of current that can flow through them is small, on the order of 3 A peak when charging and 6 A peak when discharging. On the other hand, the lithium-polymer batteries are advertised with slightly lower specific energy, but the charging/discharging rates are much higher; these rates are at 6 A for charging and 32 A for discharging.

By analyzing the specifications, the lithium-ion batteries might look more promising due to their specific energy. However, in order to power the electric motor safely while discharging at high rate, it is critical that both 4S3P (36 A) battery packs operate nominally (not isolated from the system by the protective circuitry). Otherwise, a single pack (18 A) would not be enough to power the motor. On the other hand, if a failure occurs with one of the 4S3P battery packs, the lithium-polymer chemistry is able to power the motor with only one pack plugged in to the system, thus providing desired redundancy of the EEMS.

The final decision of battery choice is based on the performance of both chemistries tested under real load conditions. A representative load experiment was performed with fully charged 4S3P battery packs equipped with PCMs to achieve a deep discharge state; the experimentally obtained plots and the detailed observations are presented later in the results and discussion section. The final result of the comparison of the performance of both battery packs shows that the lithium-polymer chemistry has higher specific energy and charging/discharging rates; therefore, the lithium-polymer batteries are the final choice for the onboard integration. Figure 38 shows the schematics of the chosen 4S3P lithium-polymer battery pack.

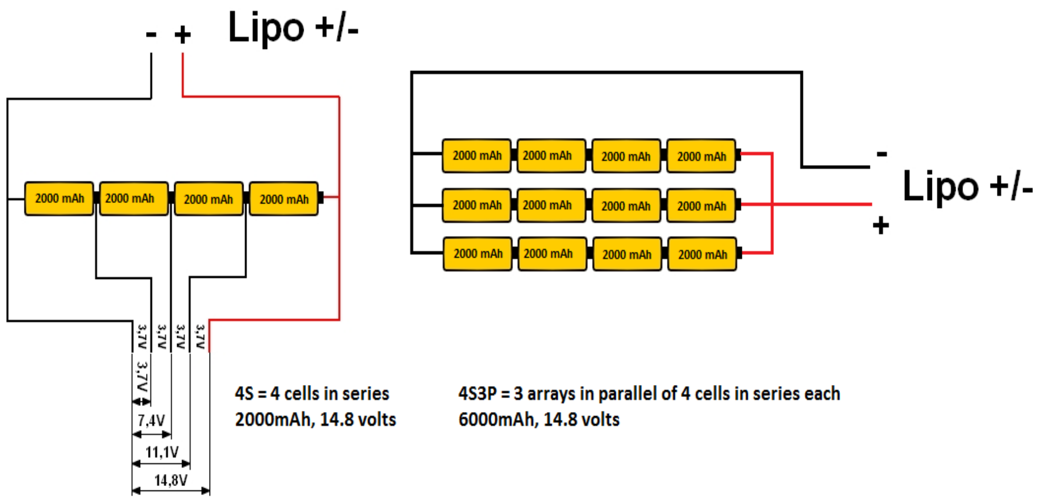


Figure 38. 4S3P lithium-polymer onboard battery pack, after [48].

A PCM circuit is the safety controller specifically designed by the industry to address one of the fundamental shortcomings of most lithium-based batteries. Namely, when the lithium-based battery is overcharged or over-discharged the composition becomes unstable and might lead to self-igniting of the pack. The functionality of PCMs prevents the batteries from leaving the safe operational region, thus preventing the fatal consequences. The EEMS architecture includes one PCM circuit per battery pack; this PCM isolates the battery pack from the electrical system should a failure occur. In summary, an advanced PCM addresses the following issues:

- overcharge – the battery is fully charged and cannot store more energy
- over-discharge – forcing deep discharge might cause permanent damage to the battery
- over current – the load demands more current than the battery is capable to provide safely
- short circuit – any short circuit in the EEMS that could affect the battery

Besides isolating the battery pack when needed, the PCM also provides continuous balancing of the state of batteries; balancing enables the individual cells in a pack to have the same voltage and receive/deliver the same current. The PCM integrated onboard the soaring UAV is presented in Figure 39.

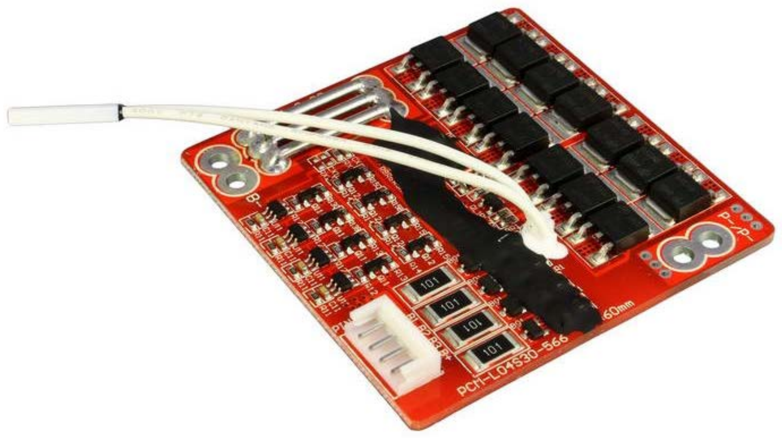


Figure 39. PCM integrated onboard, from [49].

In order to verify the protective and balancing performance of the PCM, a multiday experiment with the EEMS architecture was performed; the detailed results of experiment are presented in the results and discussion section of this chapter.

Sensing Nodes

The electric sensing nodes of the EEMS are designed to measure the analog current and voltage in all the branches; the measurements are processed by an Arduino CPU. The number and location of the sensing nodes is chosen to allow for the estimation of the amount of electrical energy produced by the PV array, consumed by the instrumentation and propulsion, and stored in the batteries. The sensing nodes are divided into two types—unidirectional and bidirectional. The unidirectional sensing nodes measure current that flows only in one direction. The bidirectional nodes are capable of measuring current in both directions. The “bidirectionality” is only required in the “storage” branches of EEMS where the current can either flow to charge the batteries or when the batteries power the avionics and propulsion system.

The design of the sensing nodes begins by selecting the sensor, which is the core of the design. There are two main types of sensors to be used in these applications: Hall-effect sensors and shunt sensors. Although accurate and bidirectional, the Hall-effect sensors are not recommended in an environment with significant inductive loads (for example, electric motors) because they can induce a magnetic field in the sensor thus resulting in erroneous measurements. On the other hand, the shunt sensor is a good choice because besides being immune to the magnetic fields, it accurately detects the change in direction of the current (“bidirectionality”). These observations were made after testing both types of sensors in a multiday experiment as shown later in this chapter.

The design of the sensing nodes is presented in Figure 40; where R_1, R_2, and R_3 are resistors used to build a voltage divider; R_S is the shunt resistor that produces the voltage drop proportional to the current measured; R_L is the output resistor; R_{OS} is the offset resistor; and C_V and C_L are the capacitors used to reject the noise in the voltage and current measurements. The specific values of the resistors vary for each sensing node. The equations required for computing the individual values are presented in (5.1).

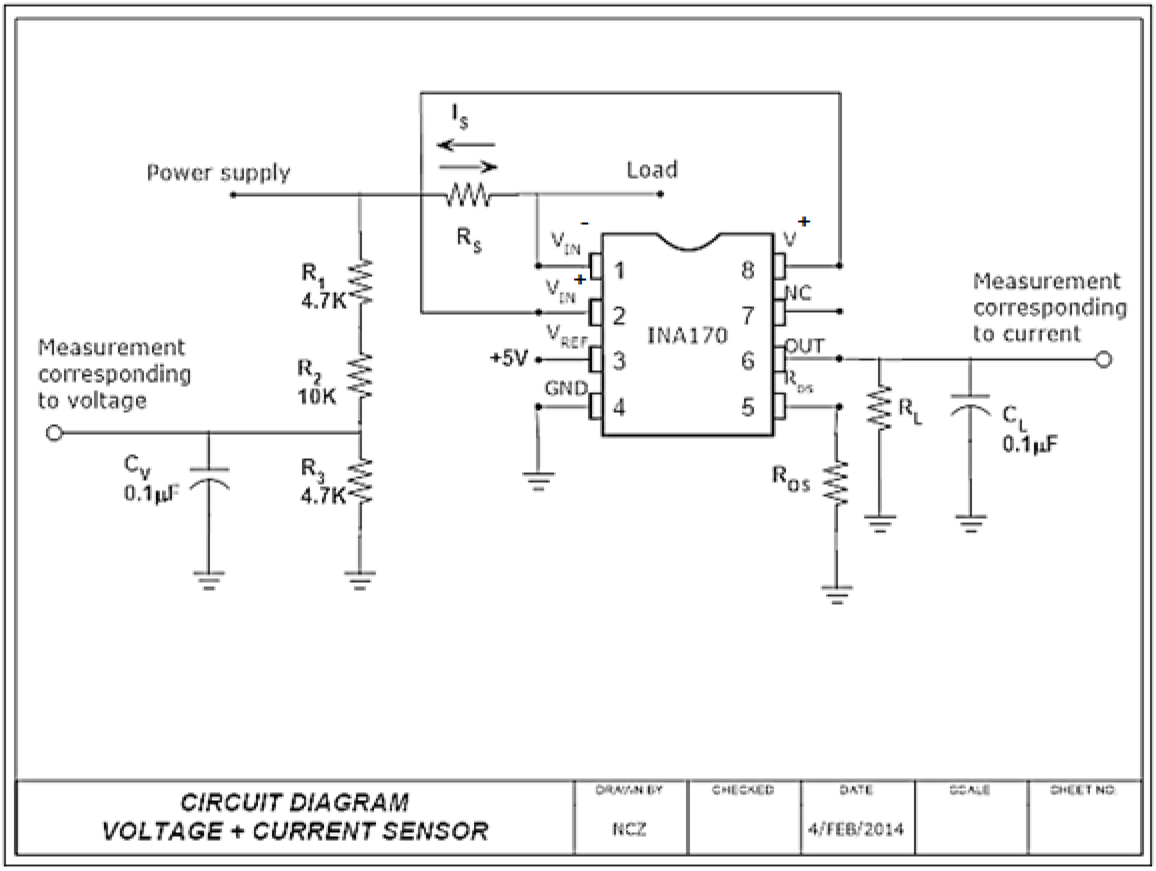


Figure 40. Design of sensing node circuit.

$$P_{S\max} = I^2_{S\max} R_S \qquad I_{S\min} = \frac{V_{S\min}}{R_S} \qquad V_{Sspan} = nI_{S\max} R_S$$

$$R_L = \frac{5volts}{V_{Sspan}} \qquad V_{\text{out unidirectional}} = (I_S)(R_S)(1000\mu Amp/Volt)(R_L) \tag{5.1}$$

$$V_{\text{out bidirectional}} = \left(\frac{V_{REF}\cdot R_L}{R_{OS}}\right) \pm \left(\frac{I_S \cdot R_S \cdot R_L}{1k\Omega}\right) \qquad R_{OS} = \frac{V_{REF}}{2.5volts}\cdot R_L$$

where $P_{S\max}$ is the maximum power across the shunt resistor caused by the maximum expected current $I_{S\max}$; $I_{S\min}$ represents the minimum current (minimum resolution) that can be read accurately as a result of the voltage $V_{S\min}$ drop (which for the case of the sensor is 0.01 V); V_{Sspan} is the voltage at the shunt resistor that determines the required R_L ($n=1$ and $n=2$ define the unidirectional and the bidirectional nodes correspondingly); and R_{OS} is the offset resistor that is used when the sensing node is bidirectional. Otherwise, the corresponding pin is not connected as well as the V_{REF} pin; V_{out} is the measured voltage that corresponds to I_S. The complete set of numerical values of the sensing nodes components is presented in Table 3; these values are computed based on the component specifications.

Sensing node	**I_{Smax} (Amperes)**	**R_S (Ohms)**	**P_{Smax} (Watts)**	**V_{Smin} (Volts)**	**I_{Smin} (Amperes)**	**V_{Sspan} (Volts)**	**R_L (kOhms)**	**R_{OS} (kOhms)**
1	6	0.01	0.36	0.01	1	0.06	80	NA
2, 3	±6	0.01	0.36	0.01	1	0.12	40	80
4, 5	40	0.001	1.6	0.01	10	0.04	100	NA

Table 3. Numerical values of the components of the sensing.

Control and Data Acquisition Unit of the EEMS

The control unit is the main data processing component of the EEMS. The unit collects and processes the analog measurements from all sensing units. This unit performs several tasks:

- receives all the analog current/voltage measurements coming from the sensing nodes
- processes the analog signals and transforms them into digital form
- computes parameters of interest, such as power and energy
- determines the aging of the batteries to predict the maximum energy they can hold
- computes the state of charge of the batteries, hence the amount of energy that exists in the system
- transmits collected and computed data over UDP
- logs all the variables into a removable flash drive

The control unit is implemented on an Arduino board, which is a very convenient platform for rapid algorithm prototyping. The Arduino board is programmed using a simplified subset of C/C++ language. The capabilities of the Arduino board include a native support for analog inputs/outputs (IO) as well as digital IO.

The aging of a battery and the state of charge can be estimated by different methods. A method named “coulomb counting” was selected to get an estimate of the energy; the method is based on the integration of the current passing through the terminals of the batteries over time. To implement the numerical integration it is necessary to know the initial condition of the battery charge state, which in a battery is never precisely known. This problem is solved by correcting the integration procedure at the moment when the battery’s state of charge gets to a known point (fully charged or fully discharged). The detailed derivation of the equations used to compute the state of charge and the aging of the batteries is presented in Appendix C.

Experiment Setup

In order to verify the EEMS architecture a prototype of the system was built and a multiday experiment was run for four days starting on October 31, 2013. Figure 41 shows the prototype of the EEMS with a solar cell array; all the components but the solar array were put inside a waterproof box to avoid problems due to humidity or rain. Only half of the solar cell array (18 out of 36) was used to match the number of solar cells used by the soaring UAV. The Hall-effect and shunt sensors were installed at different points of the system to test their performance and behavior. In order to have the 4S3P configuration of batteries, three packs of 4S lithium-ion batteries were put in parallel; each 4S pack has its own PCM unit to allow analyzing the “boundary” conditions when the protective board isolates the batteries from the system. A load of 9 W was used to simulate part of the power requirements for the avionics and payload of soaring UAV. The selection of 9 W to emulate the load was due to the fully functional autopilot.

Finally, the Arduino board was set to read, process, and log the electrical variables involved in the experiment.

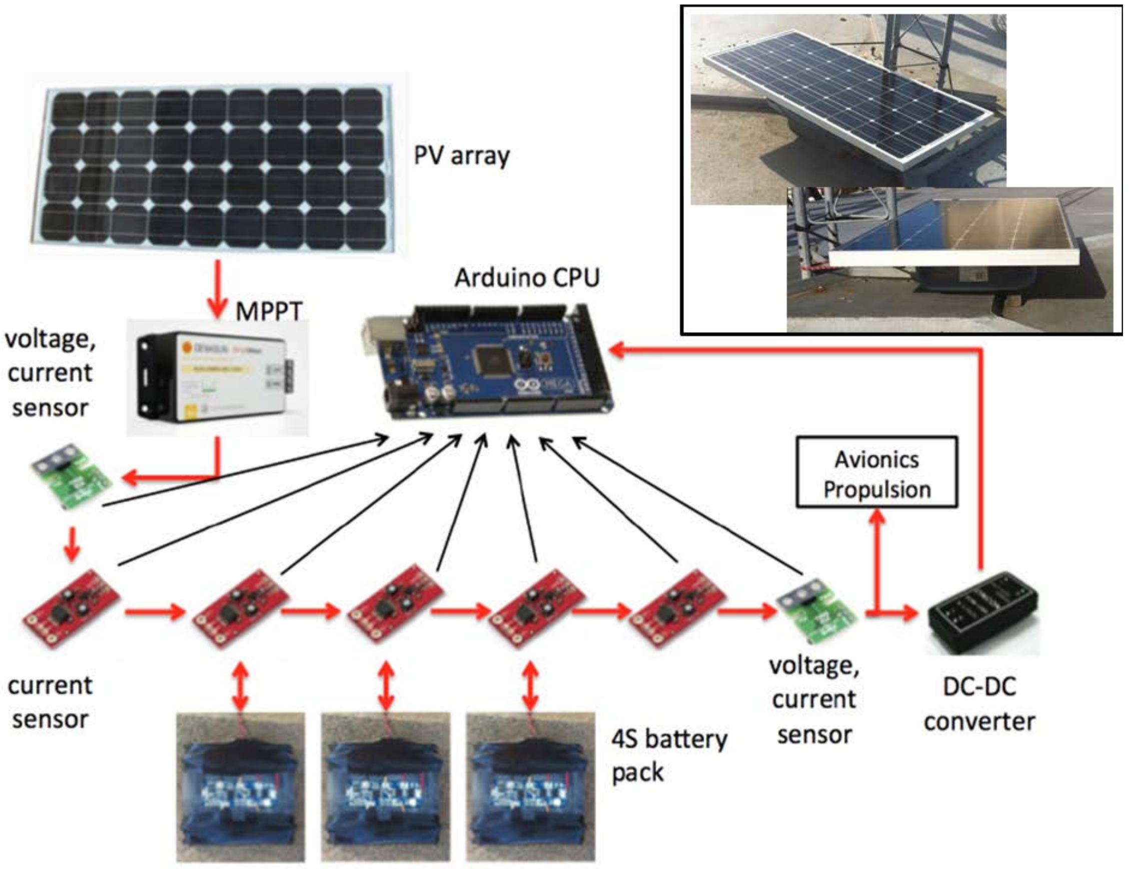


Figure 41. Hardware prototype of the EEMS installed on the roof of a building to avoid unintentional shading by structures during the daytime.

The same setup was also used in a second experiment that measured the energy stored in two different types of battery packs. Two PCM units verified the boundary states of charge of both batteries. The 12 lithium-ion battery cells in first battery pack and 12 lithium-polymer battery cells in second one had the same 4S3P configuration. Each battery pack was tested individually with a load connected to it; both battery packs were fully charged at initialization and the experiments were stopped when the PCMs isolated the battery pack at the low charge state. Conveniently installed sensor nodes integrated into the EEMS architecture allowed for estimation of the consumed power as a time integrated power, thus leading to the precise knowledge of the total energy stored in the battery packs.

C. RESULTS AND DISCUSSION

Figure 42 presents the results of the multiday experiment that tested the prototype of the EEMS. The sign convention for reading the plot is that the lines have a positive slope when energy is going into the system, while the negative slope corresponds to the energy leaving the system.

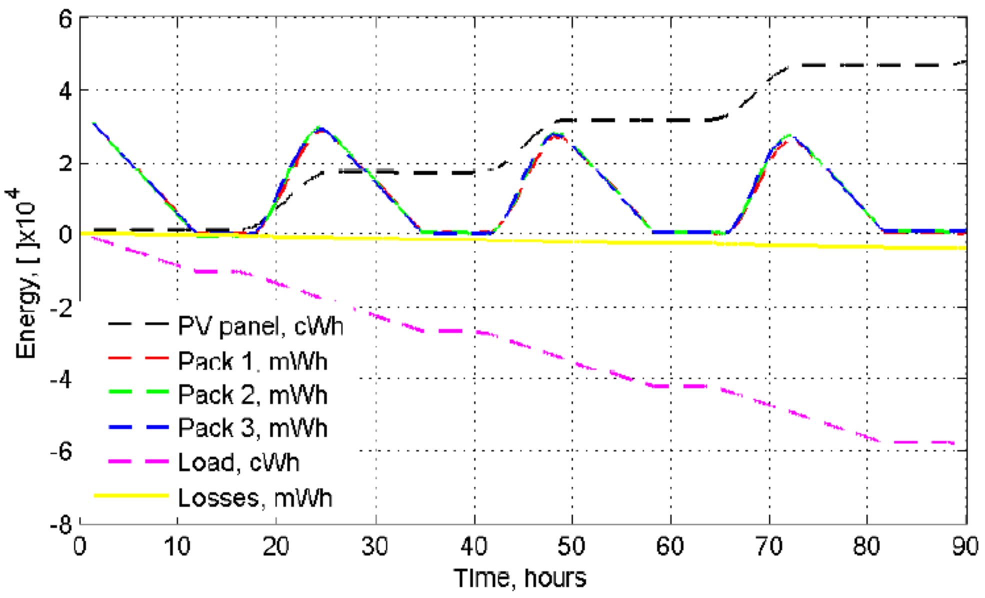


Figure 42. Result of multiday experiment with the prototype of the EEMS.

The experiment ran for 90 hours continuously with the PV cells representing the sole power source. Even though the power out of the PV panel was expected to be at 50 W (at equator under clear sun), in reality it peaked at 36 W (at the latitude of Monterey, California). As this test was performed in fall, not only were the number of daytime hours less than the number of night hours, but also the foggy weather of fall in the early mornings affected the amount of solar radiation reaching the PV cells. Overall, there is a slight difference in the amount of energy collected by the PV cells every day, which would directly affect the endurance of soaring glider. The interval of time when the PV cells provided energy to the EEMS can be characterized by the positive slope of the black line in Figure 42; in turn, the slope is zero during the nighttime.

As observed in Figure 42, the experiment started at the very end of daytime (no energy input provided by PV cells) and some nonzero charge in the batteries; therefore, the simulated load immediately started discharging the batteries. The effect of the PCMs for isolating the battery packs from the system can be explicitly observed during the nighttime. This isolation is triggered when the lower limit of the voltage is reached, thus activating the over-discharge protection. The maximum state of charge of the batteries is different for every day; that was caused by the amount of PV energy captured by the system.

The magenta curve corresponds to the dynamics of the load. It shows a constant negative slope when the load is active; the slope is zero when the batteries are “isolated” by PCM. It can be also observed that there is a minor leak of energy (see yellow line) that becomes more pronounced in a long-term operation; the leakage is caused by the losses of energy in the EEMS wiring, losses in the batteries, and some consumption of electrical energy by the sensor nodes.

The sensor that demonstrated the best performance was the shunt sensor. Therefore, the INA170 current shunt sensor manufactured by Texas Instruments is chosen for the sensing nodes of the EEMS because it provides the best accuracy with a very low level of noise.

Figure 43 shows the plots with the results of the experiment that tested the amount of energy stored in two types of battery packs: lithium-ion and lithium-polymer. Table 4 shows the comparison of the claimed end actual energy capacity of both types of batteries.

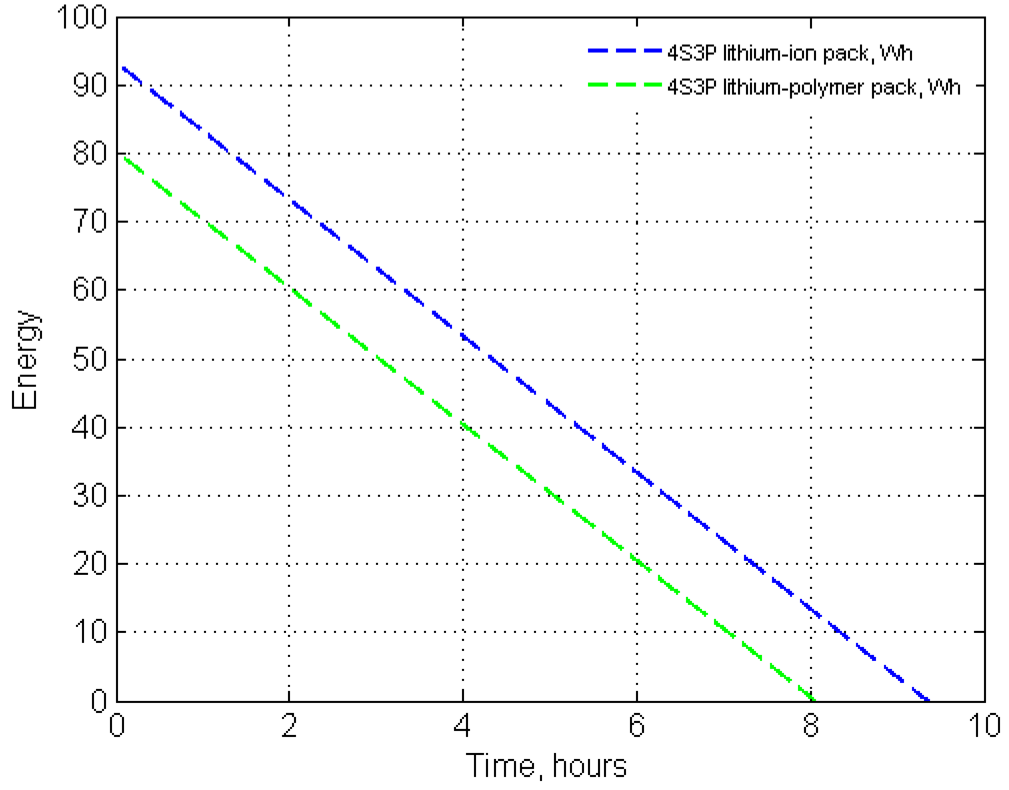


Figure 43. Performance test of energy capacity for lithium-ion and lithium-polymer 4S3P battery packs.

Battery chemistry	**Nominal voltage (volts)**	**Nominal capacity (watt-hour)**	**Actual capacity (watt-hour)**	**Weight total /casing (kilogram)**	**Specific energy (watt-hour/kilogram of total weight)**
Lithium-ion	14.8	134.1	93.2	0.640/0.04	145.625
Lithium-polymer	14.8	88.8	80.2	0.521/0.01	***153.935***

Table 4. Comparison of the energy capacity between lithium-ion and lithium-polymer 4S3P battery packs.

THIS PAGE INTENTIONALLY LEFT BLANK

VI. FLIGHT TEST RESULTS

This chapter presents the onboard implementation of developed algorithms that enable autonomous cooperative soaring and solar energy harvesting. To achieve this objective a real SB-XC glider was equipped with the hardware components discussed in the previous chapters. The scope of the experimental program covers the system identification (sink rate and the energy estimation), the thermal detection, thermal centering guidance, Bayesian search, and the TSP-based solution of the navigation task. The experimental verification of the solar harvesting wings had been performed on the ground; the process of obtaining formal Interim Flight Clearances (IFCs) is driven by the complicated NAVAIR guidelines which were not completed in time.

The platform utilized to test the algorithms is the SB-XC glider manufactured by the RnR company [50]. The cross-country soaring platform has the following characteristics: wing span = 170 in, wing area = 1545 in^2, airfoil SD-2048, aspect ratio 19.8:1, wing loading ≈ 14 oz/ft^2. Figure 43 shows the SB-XC glider used for the flight testing; there are two identical gliders and only one of them has the solar cells embedded into the wing. The SB-XC is equipped with the Piccolo Plus Autopilot [51] controlled by a Ground Control Station. The architecture of the onboard instrumentation followed the design concept of the Rapid Flight Control Prototyping [52] system that has been developed to achieve the flexibility and guarantee the rigor of experimentation.

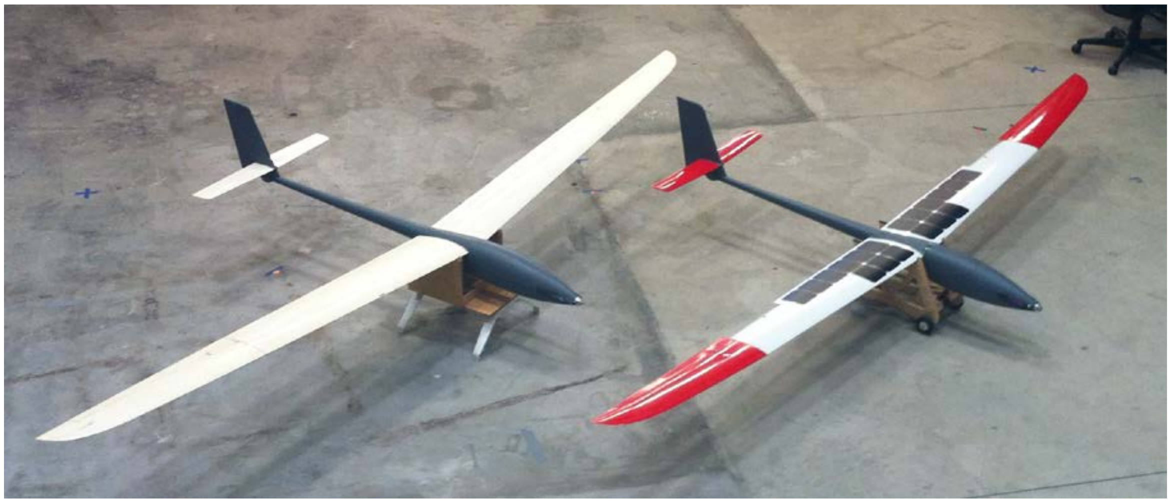


Figure 44. SB-XC gliders with and without solar panels.

The implementation of these algorithms onboard a glider is done by a secondary controller (PC104 CPU) connected to the autopilot through the full-duplex RS232 serial line. Serial communication allows reading telemetry from the autopilot sensors and submitting the low-level control commands to the autopilot at 50 Hz.

The algorithms related to the search and detection of thermals, as well as the thermalling guidance, have been designed in Simulink models and then autocoded into a real-time executable code utilizing the advanced capabilities of MATLAB Code Generator [53]; the xPC target was used as the hard real-time operating system.

The flight test was performed in the restricted airspace of Camp Roberts, California, during one of the Joint Interagency Field Exploration (JIFX) events; the objectives and the truly collaborative environment built by JIFX serve as a perfect mechanism for rapid transition of research results into a realistic operational environment [54].

The first experiment verified the performance of the Sink Polar identification algorithm. The algorithm, which is based on the Recursive Linear Least Square concept, should exhibit rapid convergence properties but requires enforced persistency of excitation condition [31]. Verification of the convergence and precision of parametric approximation is the key objective of this step. The ideal flight path of a glider in this experiment would be a sufficiently long straight line. However, due to the airspace constraints the glider was flown along four straight legs corresponding to the sides of a “narrow” rectangular path. The initial commanded true airspeed in the experiment was 14m/s, which was gradually increased by 2m/s until the glider reached the speed of 22m/s. At each commanded value of airspeed, the glider was given time for the long period transient to die. When in steady state flight the data acquisition process was started to last for at least 30 s segments, where the descend rate versus airspeed characteristics were measured at 50Hz. The characteristics of the Sink Rate Polar below 14 m/s, which is close to the stall speed of the NPS glider, were not investigated because there was minor cross wind that might have affected the accuracy of the estimator and the rigor of the experiment. The total time for this experiment was 25 minutes, and the final Sink Rate Polar curve is presented in Figure 45. The resulting Sink Rate Polar is compared against the results obtained by Edwards [32] with the same glider of slightly lighter weight. From the results, it is evident that the Sink Rate Polar obtained online closely matches the behavior of the Sink Rate Polar obtained by previous researches. The online computation of the Sink Polar in a wider range of airspeed would have been possible if the weather conditions allowed safe flight at a slower speed of ≈9m/s as Edwards did. The major achievement of this experiment is in performing the system identification experiment online (in flight) and verifying theoretically expected convergence and precision characteristics of the recursive estimator.

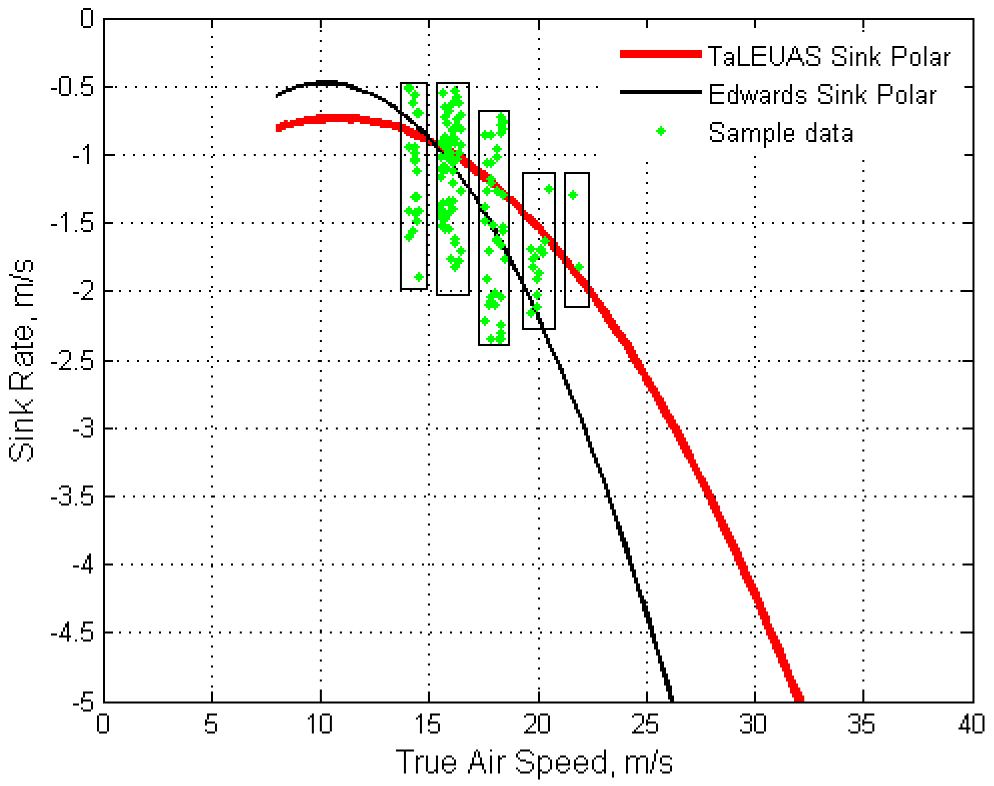


Figure 45. SB-XC's Sink Polar obtained with RLLS.

Verification of the thermal detection algorithms in a real operational environment requires tight integration of a number of algorithms, including the Bayesian search, TSP navigation, and the probability map update. During the flight the total energy approach was used as the updraft detection mechanism. The glider was guided by the heading command computed based on the suboptimal path of the TSP solution that utilized the previously given Bayesian map of the airspace in Camp Roberts. The first step in this experiment is to obtain the 3D elevation map of the Camp Roberts operational area. A rectangular area was selected and the map was automatically generated as presented in Figure 46, where the black line represents the runway. A comparison of the topography can be done by analyzing the data of Figure 47, where a view of the terrain corresponding to Camp Roberts is extracted from Google Earth.

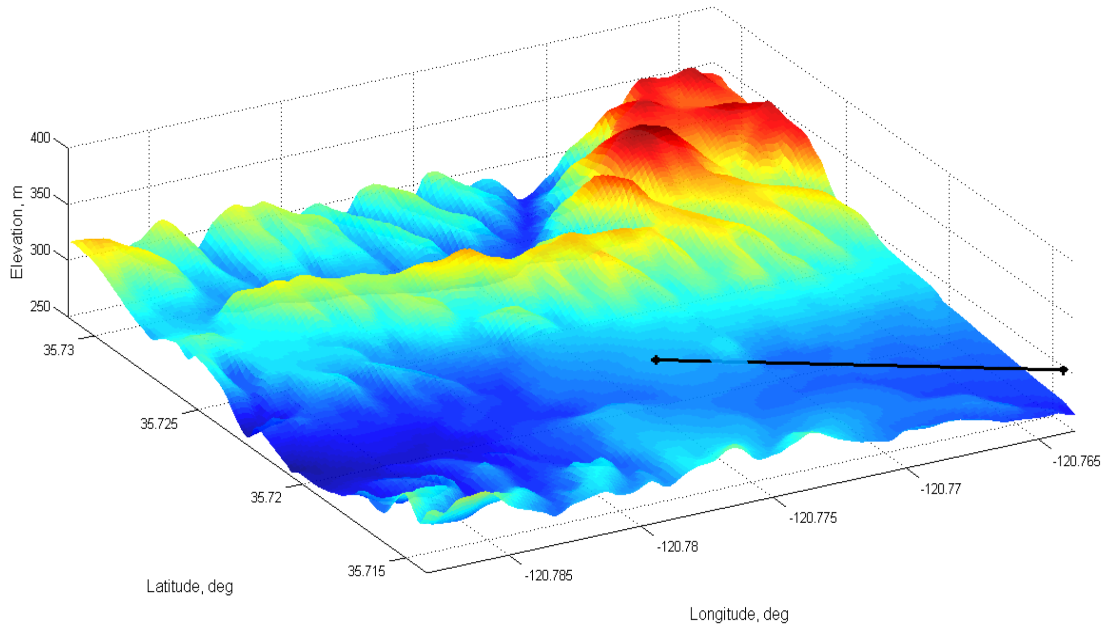


Figure 46. 3D elevation map of Camp Roberts.

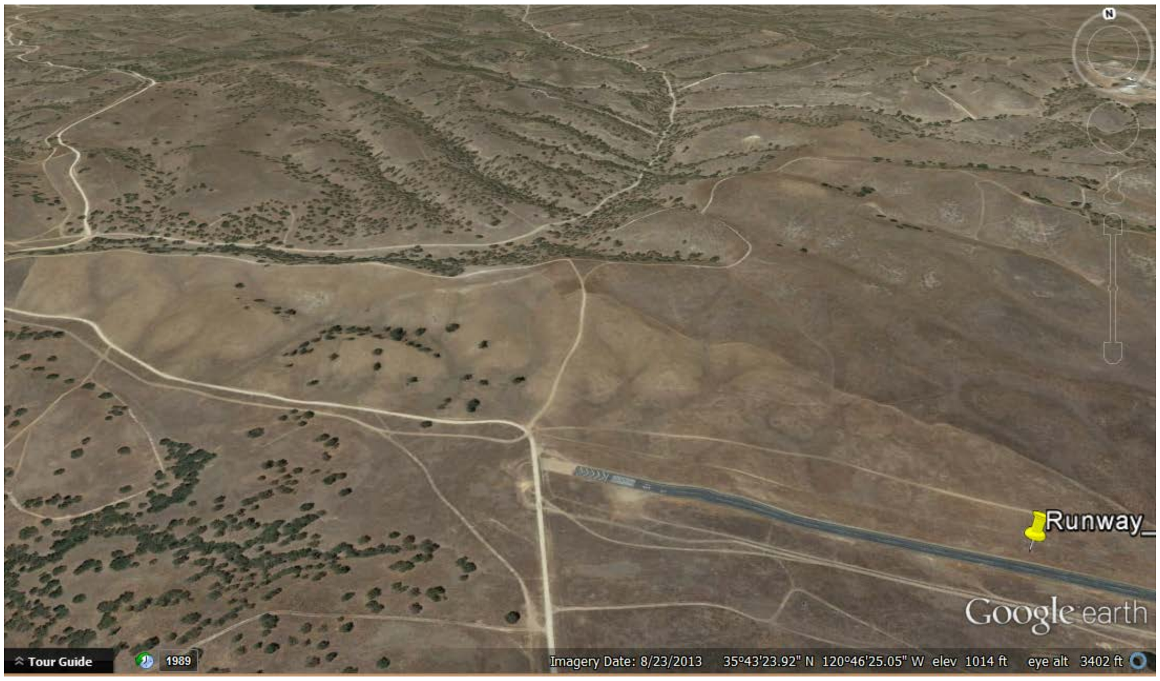

Runway_
Google earth
Tour Guide
1989
Imagery Date: 8/23/2013 35°43'23.92" N 120°46'25.05" W elev 1014 ft eye alt 3402 ft

Figure 47. Snapshot of Camp Roberts extracted from Google Earth, from [55].

The data extracted from the elevation map is then used to discretize the area of operations in square cells of ≈100m length; historical observations suggest that the thermal updrafts in this region of California have smaller diameter. The slope of the terrain and its derivative were used to determine the characteristic cells with the highest probability of finding thermals; thus, the prior probability map was generated. An example of the prior probability map is given in Figure 48.

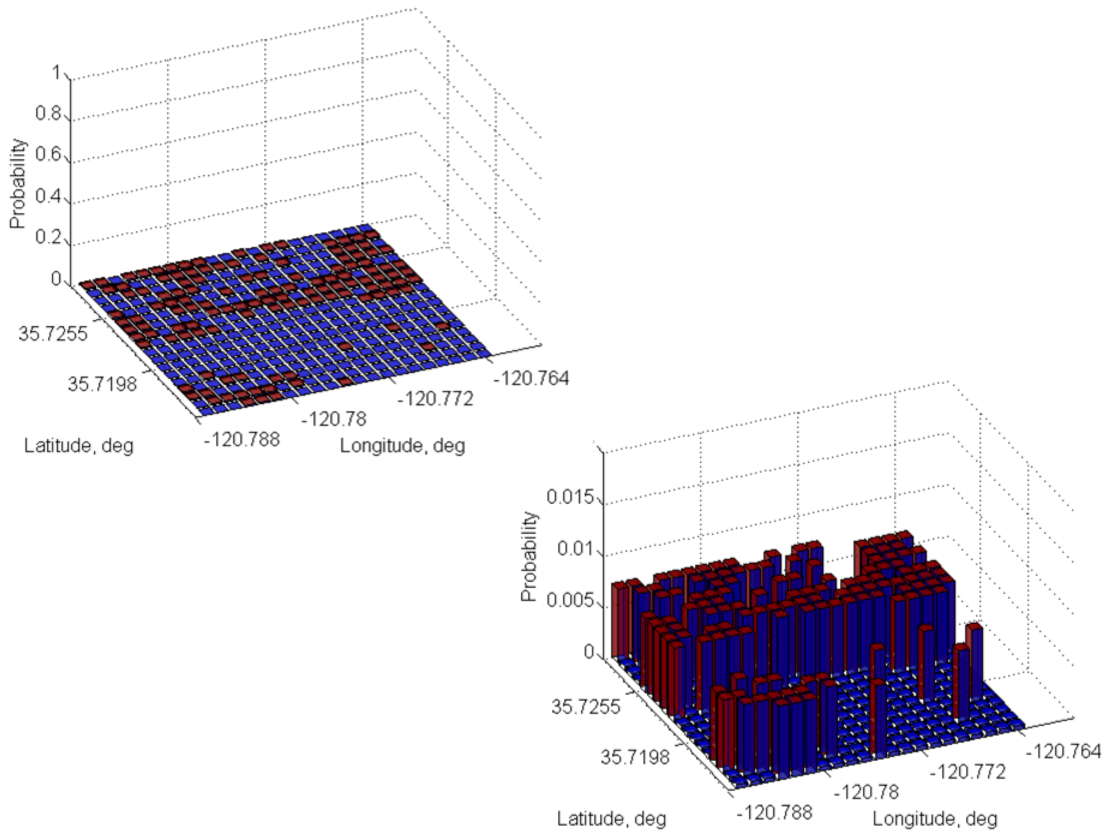


Figure 48. Prior probability map at Camp Roberts on the left. Image on the right shows its rescaled version to give sense of the magnitude of probability in each cell.

Figure 49 illustrates the solution of the TSP task projected over the local terrain. Analysis of the TSP solution shows that the trajectory travels mainly over the hills and flat lands where the updrafts are more likely to exist. Flight safety considerations and the constraints of the restricted airspace did not allow us to start the experiment at the ceiling altitude and fly through the entire path given by the solution of TSP task. Every time the glider reached the low altitude threshold (450m MSL), it was necessary to initiate climbing with the electrical motor. Given these circumstances, only a portion of the total path, shown as a red line in Figure 49, was covered.

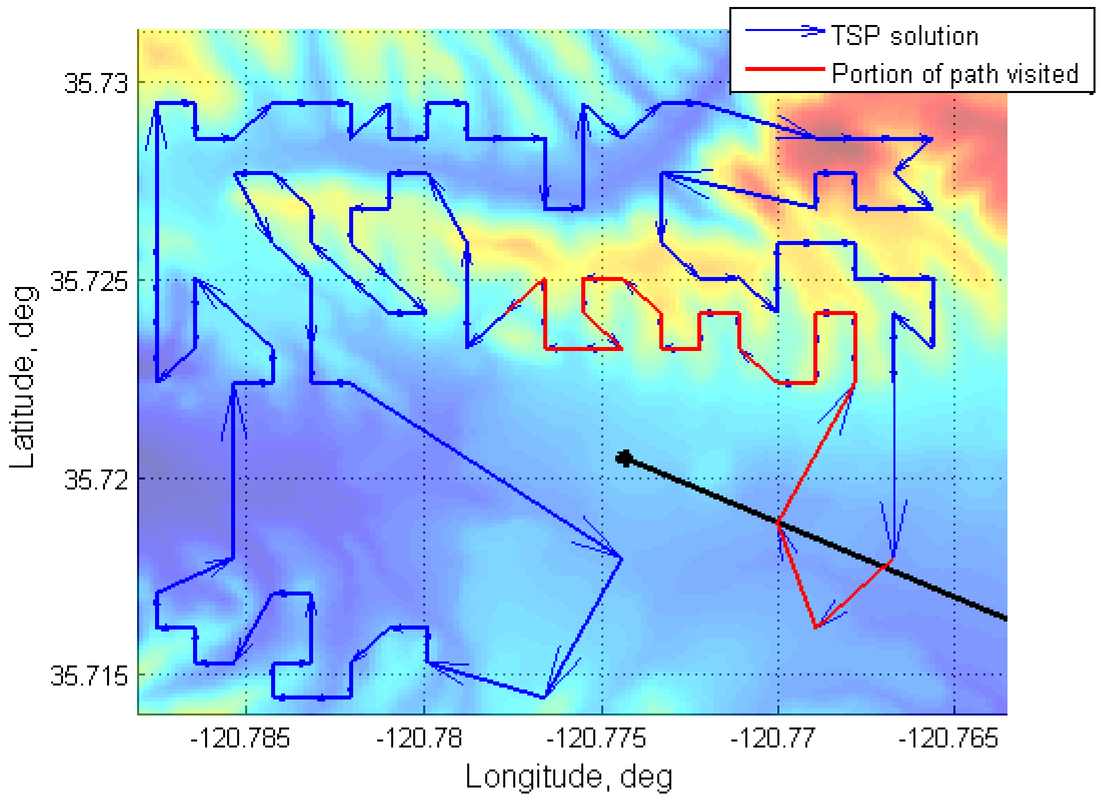


Figure 49. Path given by the solution of the TSP task and the portion of the path covered by the soaring glider in one of the experiments.

The probability map was expected to evolve with the detection of thermals; the final version of this probability map is given in Figure 50, which illustrates the updated probabilities of cells as the thermals were detected during the flight.

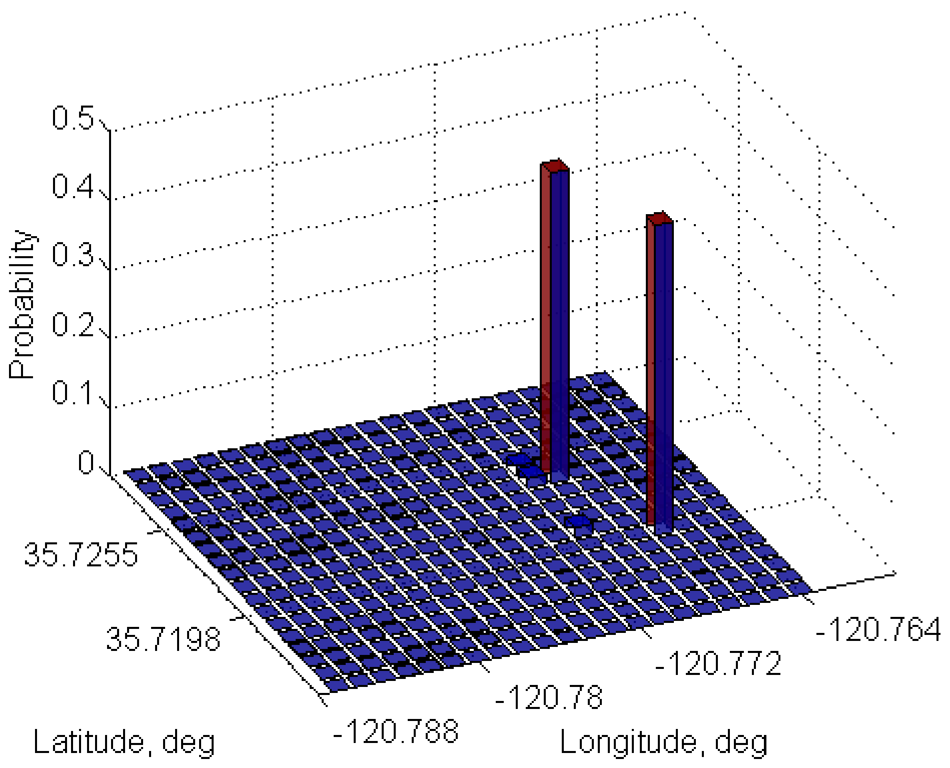


Figure 50. Probability map at the end of the experiment.

In the afternoon of the same flight experimentation day, the wind gusts were too high for safe and effective research experimentation. Therefore, the decision was made to postpone the flight experimentation with the thermal centering control. However the SIL results of the thermal centering guidance and the performance of collaboration of multiple gliders has been obtained in the integrated SIL environment; see details in Chapter III.

Since the “solar wing” was not implemented in real flight, the verification of the feasibility and performance of the onboard EEMS was performed on the ground during the same day. An experiment that tested the capabilities of the solar cells embedded into the wings to charge the 4S3P lithium-polymer battery packs was performed; the EEMS was equivalently loaded to represent the realistic operational conditions. Technically, it was necessary to fully charge a pack of batteries and verify correct operation of PCM units. Figure 51 shows the time history of energy corresponding to the elements involved in this test. First, it is worth noting that the energy of the load has a negative sign because it is leaving the system; this is the same sign convention used for the EEMS test described before. From the figure it is evident that the energy extracted from the solar cells equals the energy stored in the batteries plus the energy consumed by the load. The solar cells had a peak output power of ~50 W; however, the demand of energy for the battery pack and the load was small compared to the energy that can be obtained from the solar cells. In a short experiment like this, the losses due to wiring and other minor parasitic effects are not as explicit as in the multiday experiment described before (Chapter V, Figure 42).

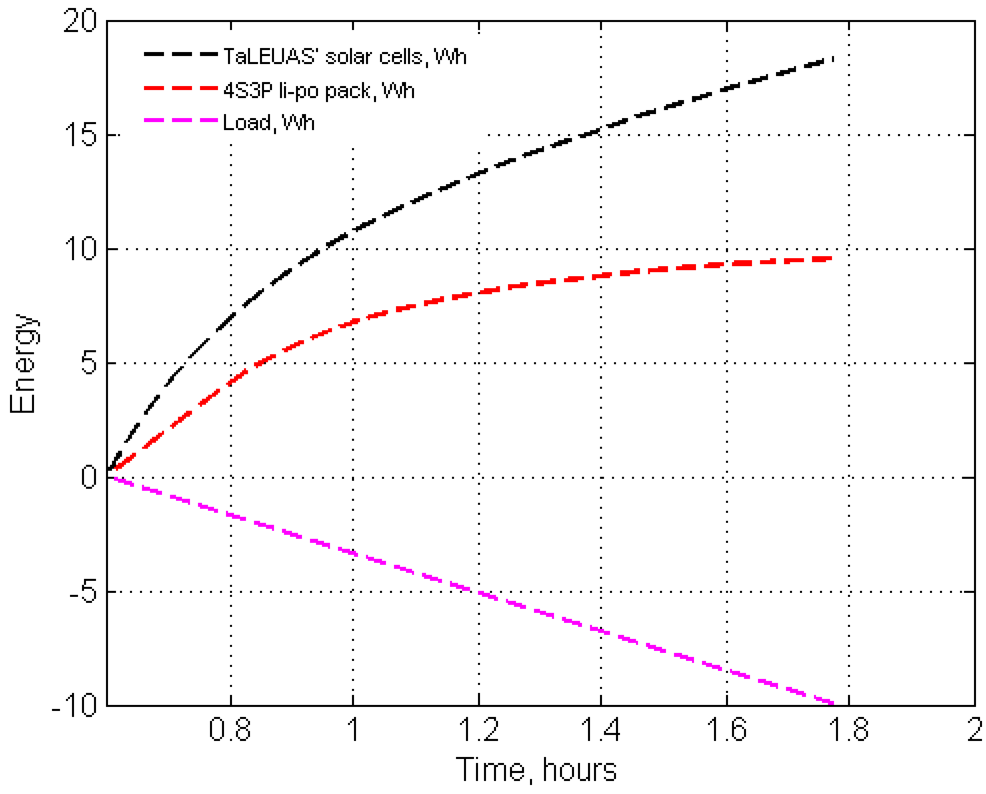


Figure 51. Energy balance and performance of solar cells powering load and charging a 4S3P lithium-polymer battery pack.

THIS PAGE INTENTIONALLY LEFT BLANK

VII. CONCLUSIONS AND FUTURE WORK

This thesis designed and implemented the software and hardware architecture to enable an autonomous glider with a hybrid energy-harvesting technique; such a technique is based on the use of convective thermals (updrafts) to gain potential energy and solar photovoltaic cells to produce electrical energy.

The research spans many disciplines such as aerodynamics, flight dynamics and control, structures and manufacturing processes, navigation and guidance, meteorology, battery technologies, solar power, communication, and electronics design.

A. CONCLUSIONS

From high fidelity simulations and flight results, it was verified that the overall process to extract energy from thermal updrafts is feasible and efficient, and requires integrating a series of algorithms such as:

- identification of the natural Sink Rate Polar of a glider to determine its nominal behavior
- detection of thermals based on the Sink Rate Polar and the total energy approaches
- optimization of the search task by means of the Bayesian search and TSP solutions

Embedding the semi-rigid photovoltaic cells into the skin of the glider wing provides a lot of advantages. Conformal shaping of solar cells and negligible (~2%) weight gain are the enablers for generating a sufficient amount of electricity from them.

This first iteration of the solar autonomously-soaring glider is a prototype to further test the achievable boundaries of the energy-harvesting technique. Even though the integrated solar-soaring architecture is capable of flying during the daytime, and the electrical energy stored onboard would only be enough to support a portion of the nighttime flight, this initial step is the fundamental achievement that provides practical verification of the scalability of future modifications. The scalability of engineering solutions needs to explore the variation of cross-correlated parameters of the system that include the aircraft geometry, the size and the architecture of the solar array, the battery size, onboard instrumentation, and utility sensors. It is clear that the scalability of the engineering design will directly impact the scalability of the mission that a flock of gliders can potentially perform.

B. FUTURE WORK

The next and most important step is to test all capabilities of the developed soaring glider in a flight test during daytime. Once the hybrid energy-harvesting technique is proven in flight, it is necessary to run an experiment to test the maximum endurance enabled by the soaring platform; the flight should start early in the morning and continue during the nighttime. Online analysis of the energy balance (consumption versus PV generation) is the most important experiment to be performed. With the results obtained from these experiments, the calculation of the desired scaling of the system (number and particular design of the solar-soaring glider) to match the desired mission requirements can be made.

One way to improve the endurance of a solar-soaring glider is by using new electronic devices that require even less power than the ones used in this current prototype; the use of batteries with a higher specific energy; and the installation of solar cells with higher efficiency.

There is yet room for improvement of the performance of the proposed thermalling controller. The lack of performance exists because the Lyapunov stability analysis typically used in these settings only provides the regions of feedback gains (k_1 and k_2) which guarantee stability of the system; performance specifications are not addressed. The variety of flight configurations (trim setting, nomenclature of controls surfaces and their configuration, weigh and balancing, etc.) of aircraft and shapes of convective updrafts require continuous retuning of feedback gains in order to achieve maximum climb rate of the glider. One approach to address this lack of performance is to implement a learning mechanism to determine the best gains for every configuration of the glider and uncertainties of the updraft; this technology is under development now by implementing the online learning and adaptive control techniques.

This thesis was only concerned about the development of the soaring glider as a platform capable of performing multiday operations. However, nothing about its application in a specific mission was analyzed. This is an area that now is being addressed in some of the control and operations research classes at the NPS Mechanical & Aerospace Engineering department. During these classes the students develop algorithms for specific missions with flocks of soaring gliders.

Estimating Cumulative Energy State for Optimal Planning of ISR Mission by Multiple Cooperative Autonomous Gliders

The operational efficiency of multiple cooperative UAVs in an ISR mission depends significantly on the optimality of mission planning. In an extended endurance ISR mission, when the UAVs are implemented by a novel class of soaring gliders capable of harvesting energy from the convective air and solar panels, the predictive methods of estimating the onboard energy become the most critical factors in planning of persistent ISR. Therefore, it might be necessary to develop a new approach to the estimation of the total onboard energy of a flock of gliders which can be used as an optimization metric during the mission planning phase. In an envisioned scenario, the ISR analyst plans a mission of multiple autonomous gliders such that either the feasibility of the desired mission is explicitly assessed, or the number of gliders and the mission “horizon” are automatically adjusted to optimally fit into the predicted value of cumulative energy resources.

On one hand, the problem of estimating the cumulative energy relies on the design of novel methods of combining various physical sources of energy: the potential energy of height and the electrical energy stored in onboard batteries. The key challenge here is in finding a formal representation of the cumulative energy state while the physical nature of its sources is different. On the other hand, the cumulative energy state needs to be manipulated to predict the possible transformation of the onboard power into, for example, the feasible operational range or the operational time on target. Therefore, it is intuitive that the cumulative energy can be used to either maximize the figure of merit (FOM) of the collaborative ISR mission or to evaluate the FOM for a mission at hand; the FOM can be specific for a particular mission and in a typical persistent ISR scenario can be represented by the time on target or the covered operational area.

Developing the approach of estimating the cumulative energy state of single and multiple soaring gliders will not only reduce the demand upon the mission planner, but will also increase the guaranteed level of the ISR mission effectiveness. The current mode of mission planning for a traditional gas- or electric-powered UAV considers the longest operational time due to onboard gas or power supplies at the most. Mission planning of multiple cooperative UAVs has just started to emerge, and is typically implemented as an ad-hoc method with no prior guarantees of mission success. These limitations motivate the development of a cumulative energy state estimation algorithm of a flock of energy-harvesting gliders to facilitate enhanced operational efficiency by accounting for energy constraints at the mission planning level.

APPENDIX A. CONDOR-SIMULINK INTERFACE

To validate and verify the algorithms developed, an interface between the high-fidelity Condor Simulator and the MathWorks Simulink was developed. Simulation allows testing the developed algorithms implemented as mathematical models. The level of similarity between the simulation and the real system depends on the complexity of the mathematical model. To simulate a soaring glider, it is necessary to have an accurate representation of the aircraft aerodynamics and flight dynamics, physical principals of sensing instruments, and atmospheric phenomena, such as formation of thermals, wind, and solar radiation. The Condor soaring simulator meets all these requirements and provides the following additional advantages:

- offers the six degrees of freedom (6DoF) models of gliders, which capture accurately the aerodynamic characteristics of the real airplanes
- provides a variety of glider models corresponding to real piloted gliders
- has available data acquisition from the simulator, sensing the states of the flight environment
- supports the mathematical model for the thermals, which predicts their behavior more accurately than any other simulator
- provides high fidelity simulation of flight and visual perception of environment, which allows training human pilots
- supports a cooperative network of gliders
- supports the performance of the algorithms developed for a glider that can be compared to the performance of human pilots
- provides “control” of the atmospheric conditions, such as characteristics of the thermals, geographic location, wind, and the solar radiation given by the date/time of the simulation
- supports repeatability of the atmospheric conditions at every simulation

The objective of building the Condor-Simulink interface is to enable data exchange between Condor and the Simulink control development environment; the information coming from Condor is used to read the states of the glider and use them to develop and test control algorithms such as autopilots, thermalling controllers, thermal detecting algorithms, Bayesian search, and any other algorithm used onboard. The outputs of these algorithms are control commands that are sent to the Condor via the same interface.

By default, it is possible to get an UDP ASCII stream of telemetry data from the simulator, which contains a subset of current states of the glider and the operational flight environment. In order to read an extended set of telemetry states—including latitude, longitude, x, y and z positions among others—from the simulator and to send back commands, it was necessary to implement an API between the Condor software and the Simulink development environment. This API was developed by Dmitrij Koniajev, Senior developer in the Demand-Side Platform team, Adform Lithuania, J. Jasinskio 16C, LT-01112 Vilnius, Lithuania, dimchansky@gmail.com. The API is installed in the host computer and some configurations further explained must be done.

Figure 52 shows a block diagram of the Condor-Simulink interface. The API receives/sends data from/to Condor; therefore in Simulink it is necessary to develop an interface that solves the following tasks:

- extract (decode) all the variables coming from the simulator through the API, convert them in physical data for further manipulation
- encode the commands for the control surfaces and send them to the Condor simulator via the API

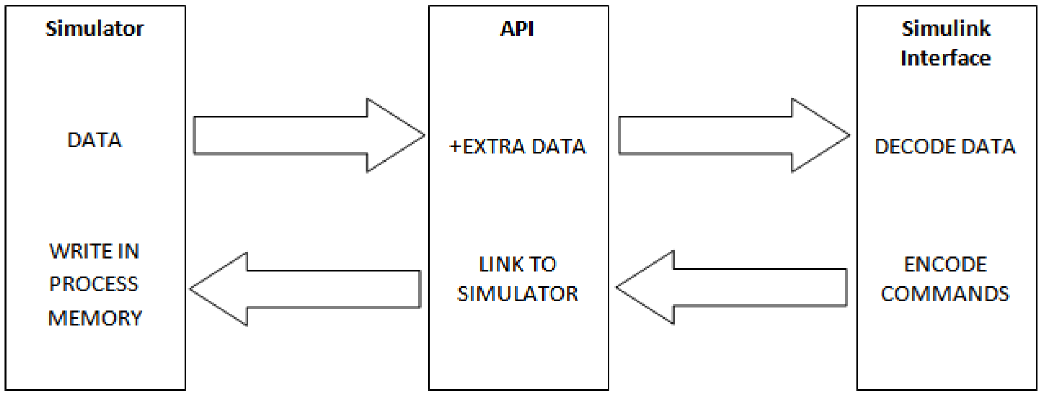
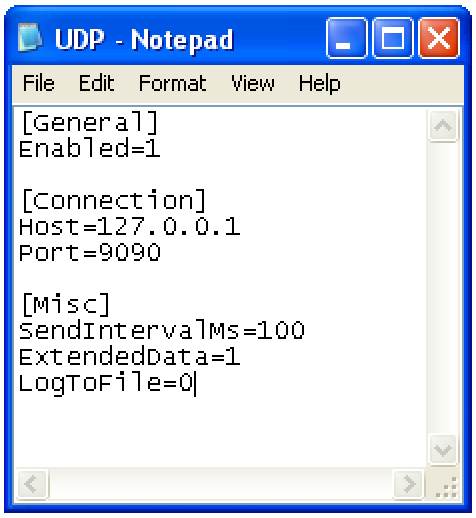


Figure 52. Block diagram of the Condor-Simulink interface.

Configuration Files of Condor and Condor API

To establish the link between the simulator, the API and Simulink; IP ports and addresses need to be set up according to a specific network configuration. Before the installation of the API, it is necessary to configure the Condor simulator for sending a default set of states; this is done by accessing the directory C:\Program Files\Condor and selecting the file UDP.ini, which contains some parameters to be set as in Figure 53.

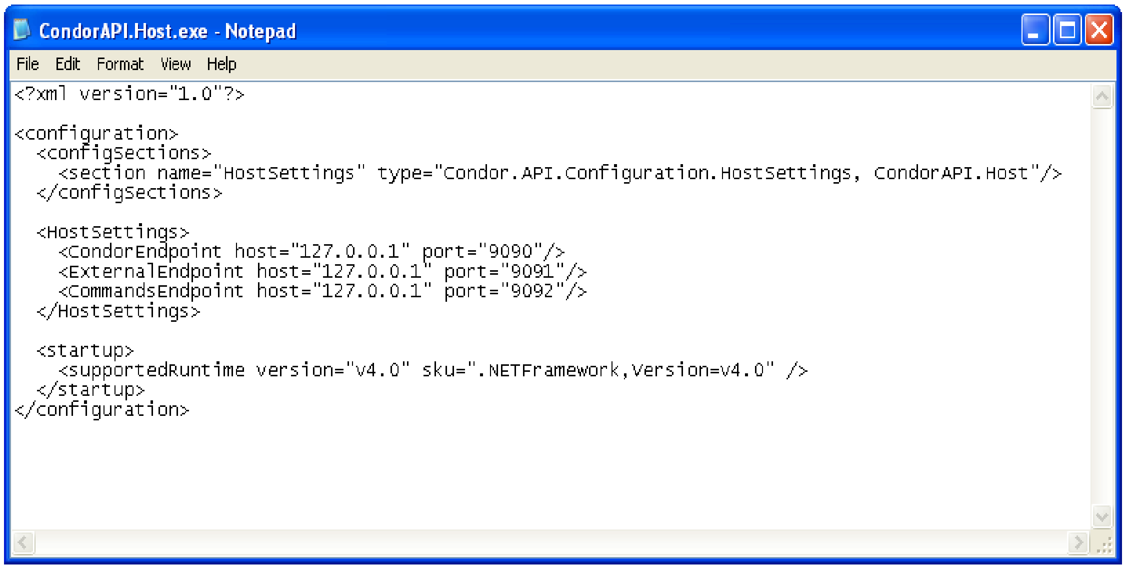


```
[General]
Enabled=1

[Connection]
Host=127.0.0.1
Port=9090

[Misc]
SendIntervalMs=100
ExtendedData=1
LogToFile=0
```

Figure 53. Customized parameters for UDP.ini file.

The “Enabled” parameter must be set to 1 in order to get the UDP stream from Condor; and the Host and port are the ones where this stream is going to be sent. Any port can be chosen within the computer that is running Condor.

The following step is to install the Condor API on the host computer where the Condor simulator is running. After the installation is complete, the file CondorAPI.Host.exe, which is located in the directory C:\Program Files\Condor\CondorAPI, must be configured with the parameters as shown in Figure 54.



```
<?xml version="1.0"?>

<configuration>
  <configSections>
    <section name="HostSettings" type="Condor.API.Configuration.HostSettings, CondorAPI.Host"/>
  </configSections>

  <HostSettings>
    <CondorEndpoint host="127.0.0.1" port="9090"/>
    <ExternalEndpoint host="127.0.0.1" port="9091"/>
    <CommandsEndpoint host="127.0.0.1" port="9092"/>
  </HostSettings>

  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.0" />
  </startup>
</configuration>
```

Figure 54. Customized parameters for CondorAPI.Host.exe file.

The Condor “Endpoint” parameters must be the same as in the UDP.ini file. The “ExternalEndpoint” port can be chosen at will. This is where the extended UDP stream is sent and is the point from which Simulink reads the data. The “CommandsEndpoint” is where Simulink sends commands and from where the API reads data to be sent to the process memory.

Implementation of the Simulink Interface

Standard Simulink library blocks are used to build the interface. The implementation is based on a sequential acquisition-decoding-encoding-sending process; the parameters described are only the ones that need to be changed from the default ones inside the Simulink blocks.

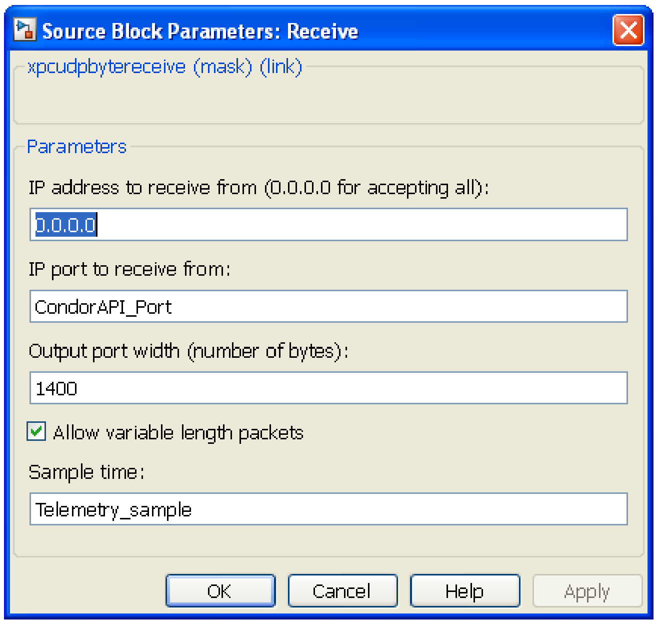
1. **Data Acquisition and Decoding.**

   In this portion of the interface, the states of the glider are taken from the Condor API and are decoded for their use in the control algorithms.

   A. **<u>Receiving Data</u>**

      a. ***UDP Receive Binary Block (Figure 55):***

         This block allows getting the UDP stream from any chosen port and IP address where it is desired to read the data.



Source Block Parameters: Receive

xpcudpbytereceive (mask) (link)

Parameters

IP address to receive from (0.0.0.0 for accepting all): 0.0.0.0

IP port to receive from: CondorAPI_Port

Output port width (number of bytes): 1400

☑ Allow variable length packets

Sample time: Telemetry_sample

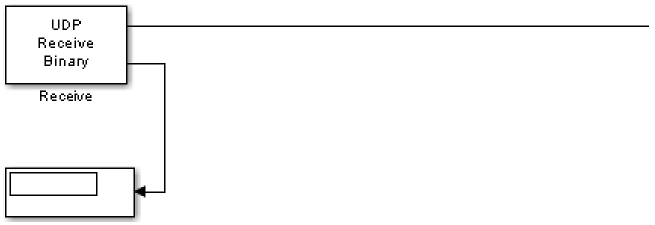


Figure 55. UDP Receive Binary block.

i. Parameters.

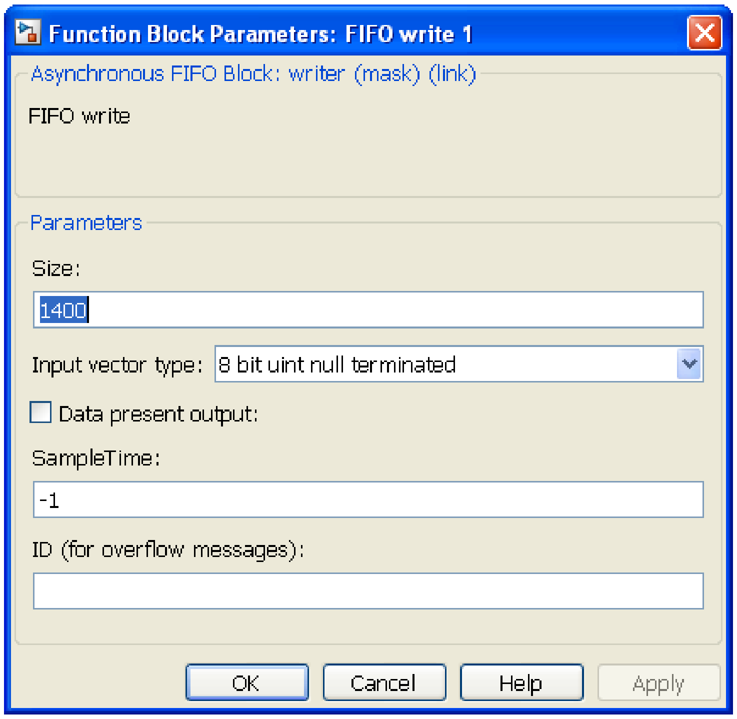
- port and IP address are the same where the API sends the data after reading it from the simulator and adding some extra variables
- the output port width is based in the maximum number of bytes that might exist in the output stream from the API. This is an important parameter to take into account to avoid misreading data and/or unexpected errors in communication. The port width must match the length of the stream in the subsequent blocks

ii. Outputs

- the upper output contains the stream and goes to the Condor API_in subsystem for the buffering process
- the second output port indicates when new data is received by the UDP Receive Binary block

B. Buffering Data (inside "Condor API_in" subsystem):

a. FIFO Write Block (Figure 56).



Function Block Parameters: FIFO write 1

Asynchronous FIFO Block: writer (mask) (link)

FIFO write

Parameters

Size:

1400

Input vector type: 8 bit uint null terminated

☐ Data present output:

SampleTime:

-1

ID (for overflow messages):

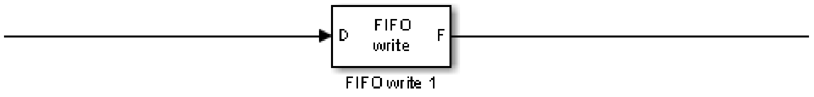


Figure 56. UDP FIFO write block.

This block takes as an input the UDP stream coming from the UDP receive Binary block and builds a serial stream to be placed into a buffer. It is necessary to implement this block before the buffer; otherwise, it will not work due to the mismatch of the types of data in the stream.

i. Parameters

- the size must match with the size established for the UDP Receive Binary block, which corresponds to a number exceeding the maximum number of bytes expected in the UDP stream

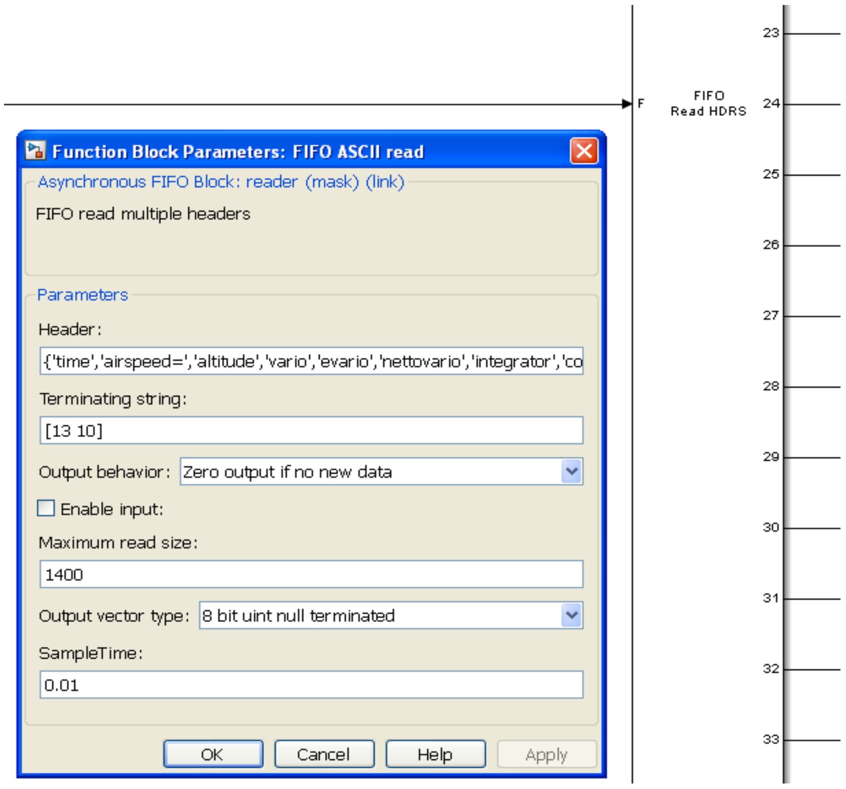
- the input vector type must be “8 bit uint null terminated” that corresponds to a byte

ii. *Output*

- the output contains the serial data that goes to the FIFO Read HDRS block

b. *FIFO Read HDRS Block (Figure 57).*

Since the telemetry data is transmitted as “name=value” pairs as ASCII symbols, the received stream coming from the FIFO write block needs to be parsed, thus separating the name of the variable from the corresponding numerical value.



Function Block Parameters: FIFO ASCII read

Asynchronous FIFO Block: reader (mask) (link)

FIFO read multiple headers

Parameters

Header:
{'time','airspeed=','altitude','vario','evario','nettovario','integrator','co

Terminating string:
[13 10]

Output behavior: Zero output if no new data

☐ Enable input:

Maximum read size:
1400

Output vector type: 8 bit uint null terminated

SampleTime:
0.01

Figure 57. UDP FIFO read HDRS block.

i. *Parameters*

- for the headers, it is necessary to type inside apostrophes the name of the variable followed by the "equals" sign; repeat this for all the variables, separating them by commas

- the terminating string for this case is [13 10], corresponding in ASCII code to “\r” and “\n”, respectively
- output behavior is “Zero output if no new data"
- the maximum read size must match with the size of the stream of the previous blocks
- the output vector type is “8 bit uint null terminated”

ii. Output

- the number of headers determines the number of outputs, and each output is composed of the bytes corresponding to the header of the variable and its value
- each output (variable) goes to an ASCII decode block

C. <u>Decoding ASCII Data (inside “Condor API_in” subsystem):</u>

a. ASCII Decode Block (Figure 58).

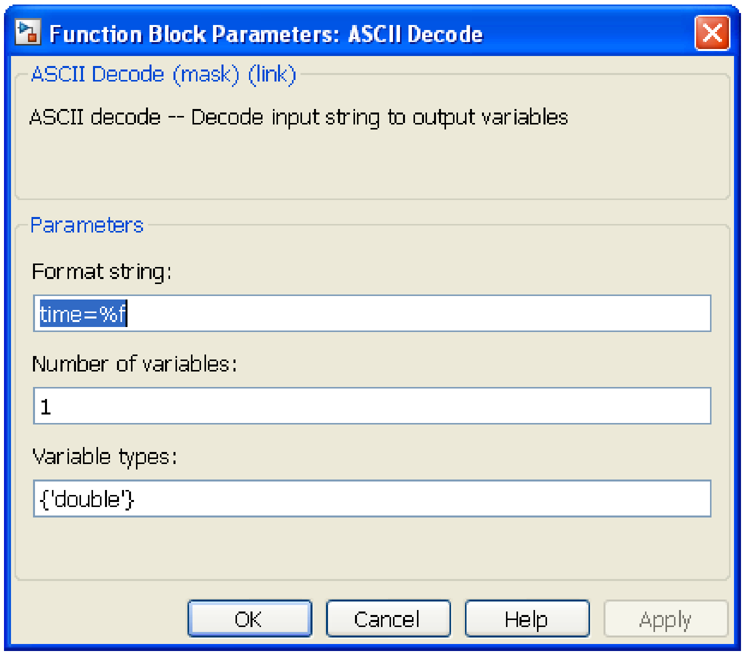
Once the bytes corresponding to the header of the variable and its value are at the output of the buffer, this block decodes the numeric value of the variable.

i. Parameters

- in the format string field, enter the name of the variable again, followed by the equals sign, after which enter the type of the variable, e.g. time=%f
- number of variables is only 1
- the variable type is double

ii. Output

- after decoding, the result is the numeric value in double type format
- after the output it is necessary to apply a scaling factor to adapt the values of the variables to a useful quantity (according to the units needed further to control the glider)
- all the variables are concentrated in a bus to use them as required


Function Block Parameters: ASCII Decode

ASCII Decode (mask) (link)

ASCII decode -- Decode input string to output variables

Parameters

Format string:

time=%f

Number of variables:

1

Variable types:

{'double'}

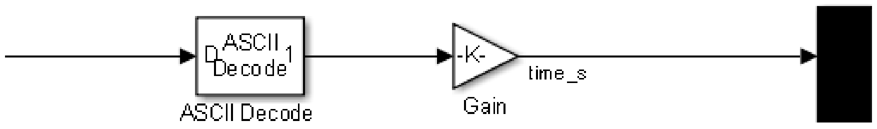


Figure 58. ASCII decoding implementation.

2. Encoding and Sending Commands

This is the section of the interface that collects, encodes, and sends the commands to control the glider, which are received by the Condor API and written directly in the process memory of Condor simulator.

A. <u>Encoding</u>

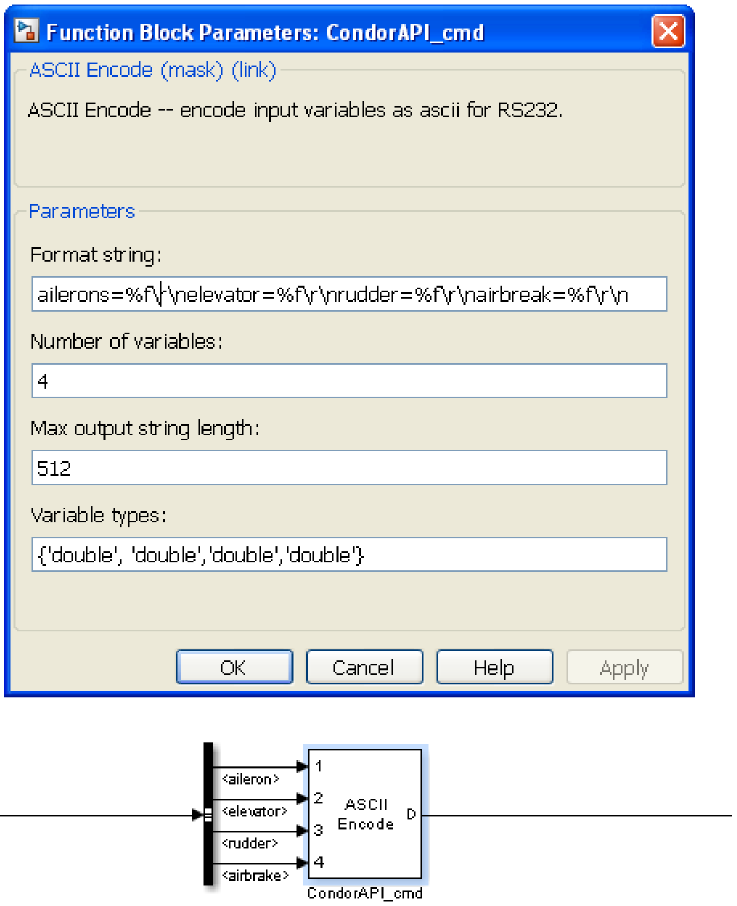
The modeled glider in Condor is controlled by deflecting four control surfaces: ailerons, rudder, elevator and airbrake. The values of the commands for such control surfaces are multiplexed/collected in a bus to use them as required.

a. ASCII Encode Block (Figure 59).

This block takes as input the values of the four commands and builds an ASCII encoded binary stream to be sent via UDP.

i. Parameters

- the format of string is as follows: “variable1=%f\r\nvariable2=%f\r\n” and so on. The “%f” determines the type of the numeric value of the variable; while the “\r” and “\n” are the terminating string characters (CR LF) needed to terminate the stream and allow the process memory of the simulator to read them unequivocally
- the number of variables is 4, matching the number of control surfaces
- the max output string length is chosen to hold the maximum number of the bytes generated by the block
- the variable types are all doubles


Function Block Parameters: CondorAPI_cmd

ASCII Encode (mask) (link)

ASCII Encode -- encode input variables as ascii for RS232.

Parameters

Format string:
ailerons=%f\r\nelevator=%f\r\nrudder=%f\r\nairbreak=%f\r\n

Number of variables:
4

Max output string length:
512

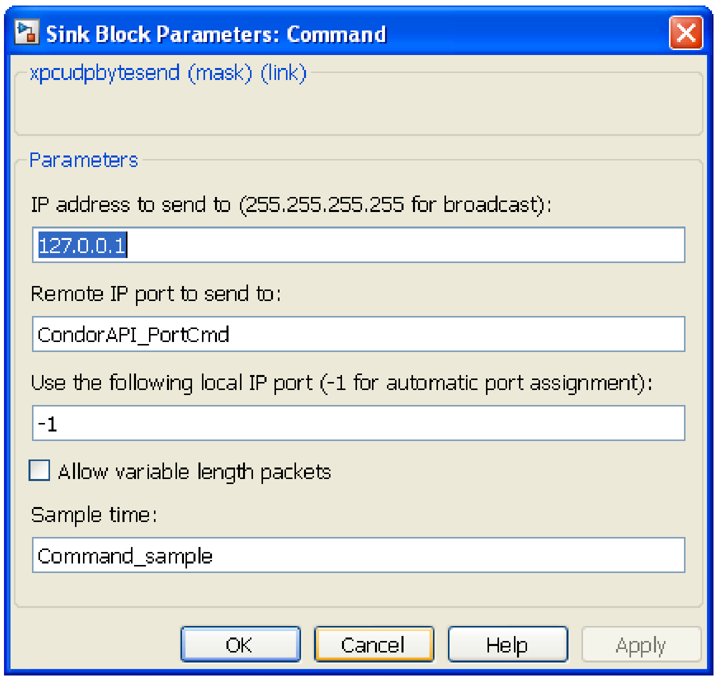
Variable types:
{'double', 'double','double','double'}

Figure 59. ASCII decoding implementation.

B. <u>Sending Commands.</u>

a. UDP Send Binary block (Figure 60):

This block sends the UDP ASCII binary stream coming from the ASCII Encode block, to any desired IP address and port.



Sink Block Parameters: Command

xpcudpbytesend (mask) (link)

Parameters

IP address to send to (255.255.255.255 for broadcast):

127.0.0.1

Remote IP port to send to:

CondorAPI_PortCmd

Use the following local IP port (-1 for automatic port assignment):

-1

[ ] Allow variable length packets

Sample time:

Command_sample

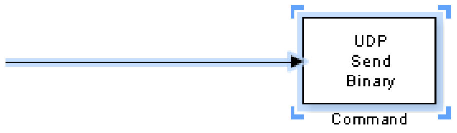


Figure 60. UDP Send Binary block.

i. Parameters

- port and IP address are the ones where the API is reading from Simulink to pass the same stream to the simulator

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B. INTEGRATION OF THIN FILM SOLAR CELLS ON TALEUAS

TaLEUAS uses a set of thin film solar cells; they were chosen because of their high efficiency (22.5%) and power/weight ratio, their thickness of 165 μm ± 40 μm, and their flexibility that allows them to be molded to the shape of the glider's wings (to be conformal). A sample of the solar cells is shown in Figure 61.

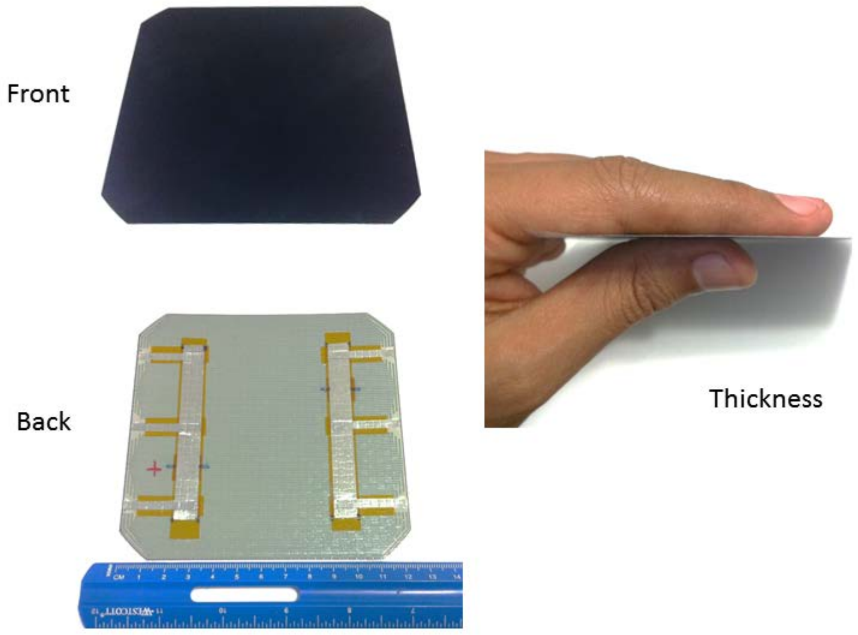


Figure 61. Thin film solar cell used on TaLEUAS.

The solar cells were embedded in the glider's wings with the goal of minimizing any parasitic aerodynamic drag that can be produced by increased roughness of the wing. The chosen technique eliminates any adverse aerodynamic effect with the implementation of a suitable manufacturing process of the wings; this process gives the opportunity of placing the solar cells in the wings' molds at the construction stage. In this way, the wings are produced with the solar cells embedded as one piece; therefore, there is no change in the shape of the airfoil and the wing. Figure 62 shows the structure of the skin of the wings and a sample section. The type of construction used by the manufacturer of the wings is a sandwich-structured composite that uses a core layer of foam and two outer layers of fiberglass. Even though the solar cells are embedded, the skin of the wings is very thin, measuring ≈ 2 mm.

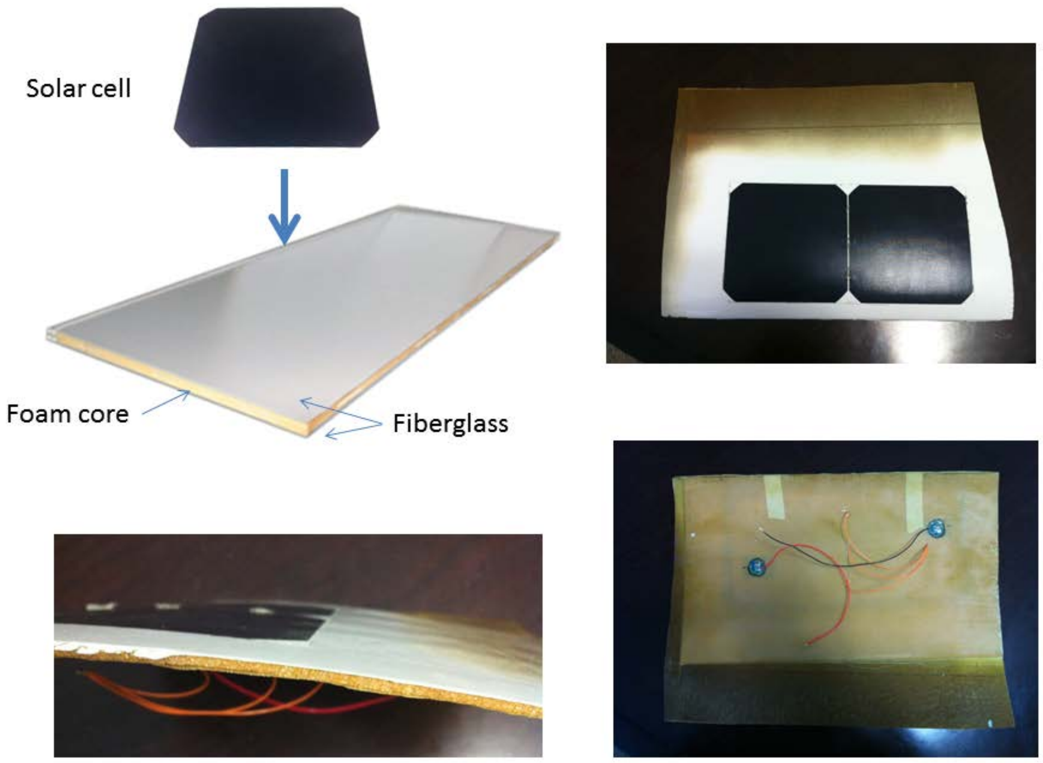


Figure 62. Structure of the skin of the wings with the solar cells embedded.

Figure 63 shows several views of TaLEUAS wings; from there it is observed how the solar cells adopt the shape of the airfoil. All 18 solar cells in the wings are connected in a series array that combines their voltage.

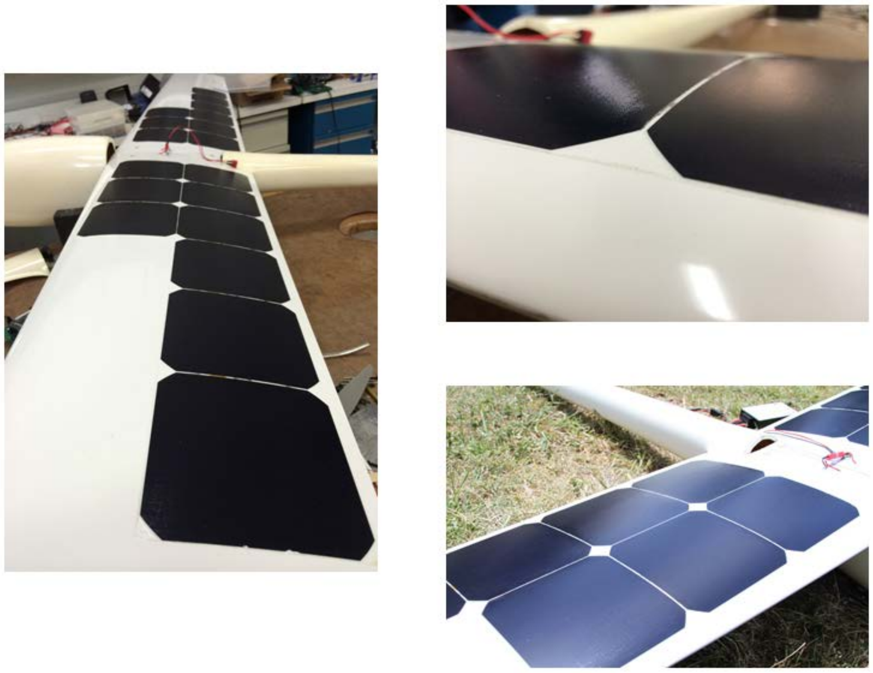


Figure 63. Solar wings.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C. STATE OF CHARGE (SOC), AGING ON BATTERIES AND CODE FOR THE CONTROL UNIT

The SOC is the amount of energy available in a battery. It can be measured as the percentage of the energy capacity of the battery, ranging from 0 to 100%; or it can be expressed in terms of ampere-hours or watt-hours. The method used to determine the SOC of a battery depends on its chemistry, the data available measured from the battery, and the fidelity desired for the estimation algorithms. Defining the SOC is not an easy task because it cannot be obtained directly from a sensor reading. Rather, a specific technique that combines measurements must be applied. To accurately determine the SOC of a battery, it is necessary to consider some parameters, such as instantaneous discharge rate, instantaneous temperature, aging, instantaneous voltage, and electrochemical mathematical model. In practice, some methods to determine the SOC are based on the electrochemical mathematical model of the battery and produce extremely accurate results; however, those models are not commonly provided by the manufacturer. Some other practical methods demonstrate good results in computing the SOC, and they are chosen because of their simplicity in implementation.

SOC Based on Voltage Measurements

Lead-acid batteries show a linear relationship between the voltage and the state of charge. This relation can be expressed as a straight line that is shifted by different factors, such as temperature, discharge rate, and aging. On the other hand, lithium batteries do not show the same behavior; instead, a small variation in voltage is observed over most of the charging/discharging cycle.

Even though the voltage is not useful to determine the SOC in lithium batteries, it is still a good indicator of the battery being fully charged or discharged. Moreover, its behavior is predictable in the neighborhood of these end points. Therefore, the voltages corresponding to fully charged and fully discharged states are used as reference states. This same approach is used by the PCM to determine when the battery pack must be isolated from the system due to subnormal operation.

SOC Based on Current Counting (Coulomb Counting)

The first requirement of this method is to have accurate measurements of the current going in and out of a battery. The current being integrated over time gives the real amount of energy stored in the battery. The accuracy in the energy estimation depends heavily on the sensing frequency to account for all the variations in the flow of current. The main inconvenience of this method is the knowledge of an accurate starting point value of the energy in the battery to begin the integration.

Method Proposed To Determine the SOC Onboard Soaring Glider

In the EEMS implemented onboard the solar glider there are sensing nodes which measure the voltage and current in all the branches, and the control unit which can determine accurately the current at the terminals of the battery at every moment. Those measurements of the voltage/current variables are very accurate given the choice of sensors used for the sensing nodes; therefore, the measurements can be confidently used to determine the SOC of the batteries.

The present design uses a combination of the current- and voltage-based methods to determine the SOC. The estimation is first determined by the coulomb counting method, which requires a known starting point to begin counting. The solution to that issue is to have the batteries fully charged at the start of a multiday flight. The control unit of the EEMS recognizes this known fully charged state (upper limit of the voltage) by measuring the voltage of the battery. If for some reason, the batteries are not fully charged at the beginning of the multiday flight, the energy counting starts at an unknown state, and as soon as the battery is fully charged given by the measurement of the voltage, the control unit corrects the value of the SOC. This approach has a behavior similar to the dead reckoning method in navigation, where the measurements from the system are used to determine the position and as soon as there is a GPS fix, the position is corrected.

The number of batteries onboard the TaLEUAS is selected such that the maximum capacity of the batteries is reached at a point during the day to survive the overnight flight. Therefore, the compensation process based on voltage should work.

Method Used To Determine the Aging of Batteries

Aging is a very important property that shortens the maximum storage capacity of the battery; therefore, it must be taken into account when the control unit determines what the SOC of the battery is. A heuristic method to determine the aging is proposed and its implementation is explained with example numbers.

The explanation is based on the characteristics of an ICR18650-30A lithium-ion battery cell. Its specifications establish that its capacity after 299 cycles $\geq$ 2030mAh (70% of the nominal capacity of 3000mAh); consequently, the aging of the battery depends on the number of cycles experienced. Each cycle is considered as the interval between the charge and the discharge of a rechargeable battery. Onboard the glider it is not possible to get a full discharge because it would make the avionics and motor stop working. Furthermore, the batteries are in a continuous state of charge and discharge, giving no chance to determine when a cycle is completed. The method proposed for determining the aging of the battery is based on the total amount of energy going in and out of the battery in a common cycle. The counting of this energy is done by the control unit and starts at zero when the battery is brand new.

Taking into account the specifications previously mentioned and assuming that the aging is linear, the history of aging plot will look as shown in Figure 64.

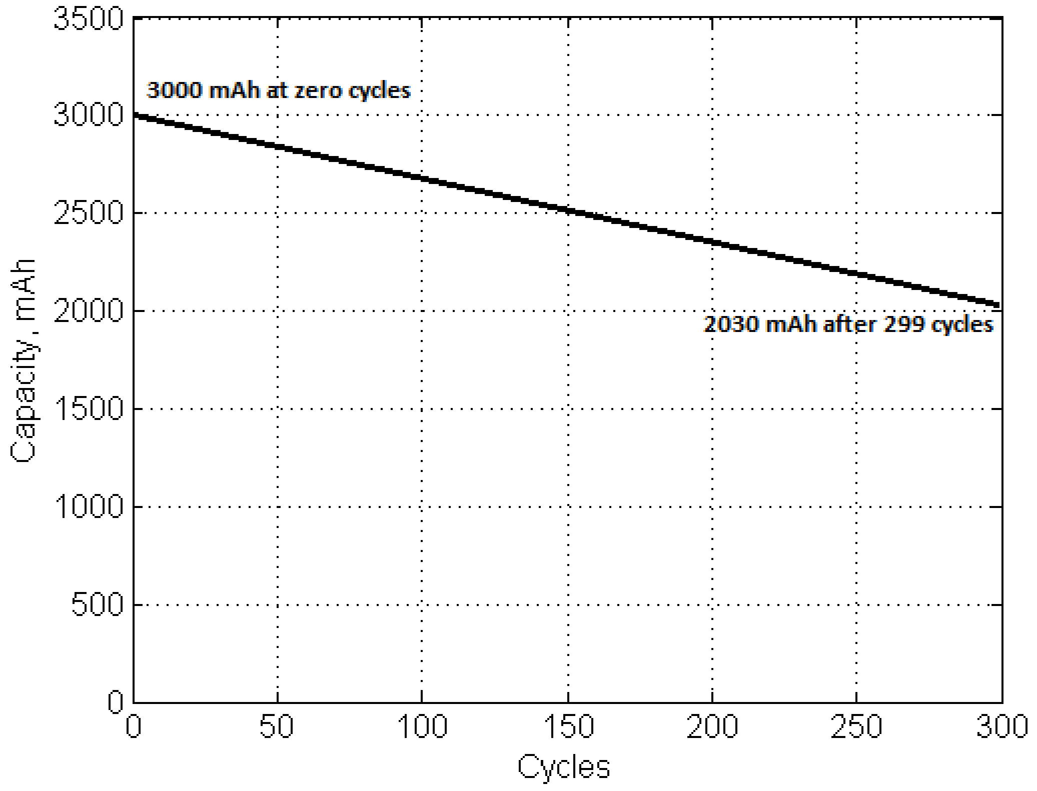


Figure 64. Aging of the ICR18650-30A lithium-ion battery cell after 299 cycles.

The equation for the aging line that joins the initial and final value of the capacity after 299 cycles is derived as follows.

$$m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{2030 - 3000}{299 - 0} = \frac{970}{299}$$
$$y = mx + b$$

Substituting the point P = (0, 3000) gives:

$$3000 = m * 0 + b$$
$$b = 3000$$
$$y = -\frac{970}{299}x + 3000$$

where x represents the number of cycles and y is the maximum capacity that the battery can hold.

One cycle is determined by putting a certain amount of energy in the battery and then taking it out. This amount of energy that the battery can hold will be decreasing as the number of cycles increases. For the continuous charging/discharging of batteries, if the energy corresponding to one cycle—twice the maximum capacity for the corresponding cycle—is measured at the terminals of the battery, then it can be assumed that one cycle has passed. This assumption can be stated as follows.

$$\int_0^x y\,d\xi = \frac{energy\,counting}{2} = \int_0^x \left(-\frac{970}{299}\xi + 3000\right)d\xi$$

$$-\frac{970}{299*2}x^2 + 3000x = \frac{energy\,counting}{2}$$

where the number of cycles “x” explicitly depends on the energy counting. Therefore, having this value it is possible to substitute in the equation of the aging line $y = -\frac{970}{299}x + 3000$ to determine the value maximum capacity that the battery can hold “y” based on the number of cycles. All these values are computed recursively by the control unit; consequently, for every computation step, the maximum capacity of the battery is known.

Arduino Code for the DAQ Unit

```
//======================================================================================
// Control_Unit_TaLEUAS
//           Date:          Feb-02-2014
//           Purpose:    Measure, process and transmit variables of the EEMS
//           Input:         Analog measurements from: voltage/current at sensing nodes
//           Output:       Via UDP all the read and processed variables.
//======================================================================================

#include <SD.h>
#include <Wire.h>
#include "RTClib.h"
#include <SPI.h>                    // needed for Arduino versions later than 0018
#include <Ethernet.h>
#include <EthernetUdp.h>   // UDP library from: bjoern@cs.stanford.edu 12/30/2008


// Digital pins to connect the LEDs that indicate the State of charge
#define greenLEDpin 2
#define redLEDpin 3

// Definition of variables for UDP communcation
const unsigned int Buffer_size = 1000;
byte mac[] = { 0x90, 0xA2, 0xDA, 0x0F, 0x23, 0xA4 };     // Enter a MAC address, this is usually printed on the board.
IPAddress ip(172, 20, 90, 210);                               // The IP address will be dependent on your local network.
unsigned int localPort = 7777;                            // local port to listen on
EthernetUDP Udp;                                              // An EthernetUDP class object to send the data over UDP
IPAddress remote_ip(255, 255, 255, 255);                  // The IP address where you will send the data:
unsigned int remote_Port = 8888;                         // remote port to send data

// Definition of variables for skyassistant
float scaling_skyassistant = (float)((4096 * 1.8 * 12.4 * 3.3 * 0.585) / 2500);
```

```
// Definition of variables for energy management
RTC_DS1307 RTC;
int DataLine = 10;
int Counter = 0;
File dataFile;
long milli_watts_sec_1 = 0;
long milli_watts_sec_2 = 0;
long milli_watts_sec_3 = 0;
long milli_watts_sec_4 = 0;
long milli_watts_sec_5 = 0;
long milli_watts_sec_6 = 0;
long MillSec_last = 0;
long MillSec = 0;
long max_capacity_p1 = 0;
long max_capacity_p2 = 0;
long max_capacity_p3 = 0;
long energy_counting_p1_mWs = 0;
long energy_counting_p2_mWs = 0;
long energy_counting_p3_mWs = 0;
float max_energy_0_cycles = 35000;                              // Nominal is 3000 mAh * 16.5 volts = 49.5 Wh. The rated is 35
                                                                // Wh = 35000 mWh
float max_energy_299_cycles = max_energy_0_cycles - 970 * 16.5;   // Nominal is 2030 mAh * 16.5 volts = 33.495 Wh.
// The consideration will be to have the same decay (3000 mAh - 2030 mAh) = 970 mAh over the 299 cycles
float energy_decay_slope = (max_energy_299_cycles - max_energy_0_cycles) / 299;
long energy_pack1_corrected_mWs = 0;
long energy_pack2_corrected_mWs = 0;
long energy_pack3_corrected_mWs = 0;


void setup()
{

  // Initialize communication with SD card
  Serial.begin(9600);                            // This is the correct baudage for Arduino Mega

  pinMode(4, OUTPUT);                      // set the SS pin as an output (necessary!)
  digitalWrite(4, HIGH);                   // turn off the SD
  pinMode(10, OUTPUT);                     // set the SS pin as an output (necessary!)
  digitalWrite(10, HIGH);                 // but turn off the W5100 chip!
  pinMode(53, OUTPUT);                     // set the SS pin as an output (necessary!)

  Serial.print("\nInitializing SD card...");
  if (!SD.begin(10, 11, 12, 13)) { // Must keep the 10,11,12,13 numbers for Arduino Mega
    Serial.print("\nError in reading SD card");
    return;
  }
  Serial.println("card initialized.");


  // Initialize the ethernet and UDP communication
  Ethernet.begin(mac, ip);
  Udp.begin(localPort);


  // connect to RTC
  Wire.begin();
  if (!RTC.begin()) {
    Serial.println("RTC failed");
  }// if

  // Assign values for the energy counting variables
  if (!SD.exists("datalog.txt")) {
    // If file doesn't exist then assign initial value for the energy counting. Could be initialized to zero.
    energy_counting_p1_mWs = 0;
    energy_counting_p2_mWs = 0;
    energy_counting_p3_mWs = 0;
  }// if
```

```
else {
  // Get the aging from the last file in the SD card

  // Open the file and go at the end of it
  dataFile = SD.open("datalog.txt", FILE_WRITE);

  // IF dataFile is open/created get the aging from datalog.txt file
  if (dataFile) {

    // Declare variables to be used
    String line = "";      // This is the buffer for the variables from the file coming as strings
    unsigned long pointer = dataFile.position();      // Pointer to read the file.
    boolean not_done = true;                          // Determines qhen all aging variables are extracted
    int var_counter = 0;                              // Counts for the number of battery packs in the system
    String aging_str[3];                              // Contains the aging as strings
    int aging_p1 = 0;                                 // Value of aging for pack 1
    int aging_p2 = 0;                                 // Value of aging for pack 2
    int aging_p3 = 1;                                 // Value of aging for pack 3


    // Extract the aging from SD card
    while (not_done)
    {

      pointer -= 1;    // Pointer will avoid the \t character
      dataFile.seek(pointer);    // Search where the pointer says
      char new_char = dataFile.peek();    // Extracts the char from the pointed byte

      // Appends the chars backwards till get a \t character
      if (new_char == '-' || isdigit(new_char)) {
        line += new_char;
      } // if
      // When hit a \t character, the program puts the characters in the correct order
      if (new_char == '\t') {
        line.trim();

        for (int i = 0; i < line.length(); i++) {
          aging_str[var_counter] += line[line.length() - 1 - i];
        } // for

        var_counter += 1;
        line = "";
        if (var_counter >= 3) {   // Accounts for the number of battery packs and exits the program when reached the
                                  // number
          not_done = false;
        } // if
      } // if
    } // while

    // Convert the strings of characters to integers
    aging_p1 = aging_str[2].toInt();
    aging_p2 = aging_str[1].toInt();
    aging_p3 = aging_str[0].toInt();

    // Convert the integers to long and to the units of mWs
    energy_counting_p1_mWs = (long(aging_p1)) * (60 * 60);
    energy_counting_p2_mWs = (long(aging_p2)) * (60 * 60);
    energy_counting_p3_mWs = (long(aging_p3)) * (60 * 60);

    // Close the datalog.txt file
    dataFile.close();

  }// if
  else {          // if the file isn't open, pop up an error:
    Serial.println("error opening datalog.txt");
    Serial.println("Must assign the aging factor manually");

  } // else
  } // else



  // Writing the header file to put data in context
  HeaderFile();

  // Set up the Digital I/O 2 and 3 as outputs
  pinMode(redLEDpin, OUTPUT);
  pinMode(greenLEDpin, OUTPUT);


} // void setup



void loop() {

  if (Serial.available() > 0) {
    return;
  } // if


  // Create the process to compute the rate of climb from skyassistant
  // Read the analog voltage in value from 0 to 1023 :
  int skyassistant_sensorValue = analogRead(A7);
  // Map it to the range of the analog voltage that in arduino corresponds from 0 to 5000 milli volts:
  int millivolts_skyassistant = map(skyassistant_sensorValue, 0, 1023, 0, 5000);
  // As the voltage beind read can only have values from 0 to 2500 milli volts, map it in skyassistant units in the range of –
  //2048 to 2048:
  float units_skyassistant = (float)map(millivolts_skyassistant, 0, 2500, -2048, 2048);
  // Compute the rate of climb in meters per second from the skyassistant units; this equation was provided by the
  // manufacturers:
  float rate_of_climb_m_s = (29.0 / (1295.0)) * millivolts_skyassistant - 29.0;  //Here calibrate with the voltage readings at
  // zero climbing from the skyassistant. Maximum rate of climb is 29 m/s
  // Create some variables to be used in the energy management
  String dataString = "";
  MillSec_last = MillSec;
  MillSec = millis();
  long DeltaT = (MillSec - MillSec_last);


  // Get measurements at solar panel check point 1
  // Read voltage from analog pin:
  int voltage_sensor_1 = analogRead(A9);
  // Convert the analog reading (which goes from 0 - 1023) to a voltage (0 - 5V):
  long milli_volts_1 = voltage_sensor_1 * (5.0 / 1023.0) * 1000 * (1000 / 245.2);
  // Read voltage to get current from analog pin:
  int current_sensor_1 = analogRead(A8);
  // Convert the analog reading (which goes from 0 - 1023) to a voltage (0 - 5V) and multiply it by the scaling factor:
  long milli_amps_1 = (current_sensor_1 * (5.0 / 1023.0) * 1000) * (1000 / 50); // Calibrate as required
  long milli_watts_1 = milli_volts_1 * milli_amps_1 / 1000;
  milli_watts_sec_1 = milli_watts_1 * DeltaT / (1000) + milli_watts_sec_1;
  long milli_watt_hour_1 = milli_watts_sec_1 / (60 * 60);


  // Get measurements at solar panel check point 2
  // Read voltage to get current from analog pin:
  int sensor_current_2 = analogRead(A10);
  // Convert the analog reading (which goes from 0 - 1023) to a voltage (0 - 5V) and multiply it by the scaling factor:
  long milli_amps_2 = ((sensor_current_2 * (5.0 / 1023.0) * 1000) - 4071.58658) / (-0.738281126);
  long milli_watts_2 = milli_volts_1 * milli_amps_2 / 1000;
  milli_watts_sec_2 = milli_watts_2 * DeltaT / (1000) + milli_watts_sec_2;
  long milli_watt_hour_2 = milli_watts_sec_2 / (60 * 60);


  // Get measurements at Battery Pack 1 (check point 3)
  // Read voltage to get current from analog pin:
  int sensor_current_3 = analogRead(A11);
  // Convert the analog reading (which goes from 0 - 1023) to a voltage (0 - 5V) and multiply it by the scaling factor:
  long milli_amps_3 = ((sensor_current_3 * (5.0 / 1023.0) * 1000) - 4119.67316) / (0.73571); // (Vmeas -Vref)/sensitivity
  long milli_watts_3 = milli_volts_1 * milli_amps_3 / 1000;
  milli_watts_sec_3 = milli_watts_3 * DeltaT / (1000) + milli_watts_sec_3;
  long milli_watt_hour_3 = milli_watts_sec_3 / (60 * 60);
  energy_counting_p1_mWs = abs(milli_watts_3 * DeltaT / (1000)) + energy_counting_p1_mWs;  // All energy
  // cummulated since the battery started working. Measured in milli watt seconds
  long energy_counting_p1_mWh = energy_counting_p1_mWs / (60 * 60);


  // Get measurements at Battery Pack 2 (check point 4)
  // Read voltage to get current from analog pin:
  int sensor_current_4 = analogRead(A12);
  // Convert the analog reading (which goes from 0 - 1023) to a voltage (0 - 5V) and multiply it by the scaling factor:
  long milli_amps_4 = ((sensor_current_4 * (5.0 / 1023.0) * 1000) - 4155.619048) / (0.689796); // (Vmeas -Vref)/sensitivity
  long milli_watts_4 = milli_volts_1 * milli_amps_4 / 1000;
  milli_watts_sec_4 = milli_watts_4 * DeltaT / (1000) + milli_watts_sec_4;
  long milli_watt_hour_4 = milli_watts_sec_4 / (60 * 60);
  energy_counting_p2_mWs = abs(milli_watts_4 * DeltaT / (1000)) + energy_counting_p2_mWs;  // All energy
  // cummulated since the battery started working. Measured in milli watt seconds
  long energy_counting_p2_mWh = energy_counting_p2_mWs / (60 * 60);


  // Get measurements at Battery Pack 3 (check point 5)
  // Read voltage to get current from analog pin:
  int sensor_current_5 = analogRead(A13);
  // Convert the analog reading (which goes from 0 - 1023) to a voltage (0 - 5V) and multiply it by the scaling factor:
  long milli_amps_5 = ((sensor_current_5 * (5.0 / 1023.0) * 1000) - 4147.101732) / (0.678409); // (Vmeas -Vref)/sensitivity
  long milli_watts_5 = milli_volts_1 * milli_amps_5 / 1000;
  milli_watts_sec_5 = milli_watts_5 * DeltaT / (1000) + milli_watts_sec_5;
  long milli_watt_hour_5 = milli_watts_sec_5 / (60 * 60);
  energy_counting_p3_mWs = abs(milli_watts_5 * DeltaT / (1000)) + energy_counting_p3_mWs;  // All energy
  // cummulated since the battery started working. Measured in milli watt seconds
  long energy_counting_p3_mWh = energy_counting_p3_mWs / (60 * 60);


  // Get voltage measurement for the load (check point 7)
  // Read voltage from analog pin:
  int voltage_sensor_7 = analogRead(A15);
  // Convert the analog reading (which goes from 0 - 1023) to a voltage (0 - 5V):
  long milli_volts_2 = voltage_sensor_7 * (5.0 / 1023.0) * 1000 * (1000 / 246.9616);


  // Get measurements at Load (check point 6)
  // Read voltage to get current from analog pin:
  int sensor_current_6 = analogRead(A14);
  // Convert the analog reading (which goes from 0 - 1023) to a voltage (0 - 5V) and multiply it by the scaling factor:
  long milli_amps_6 = ((sensor_current_6 * (5.0 / 1023.0) * 1000) - 253.590909) / (-0.815091569);
  long milli_watts_6 = milli_volts_2 * milli_amps_6 / 1000;
  milli_watts_sec_6 = milli_watts_6 * DeltaT / (1000) + milli_watts_sec_6;
  long milli_watt_hour_6 = milli_watts_sec_6 / (60 * 60);



  // Compute the equivalency in cycles based on the energy counting
  float equivalency_cycles = (-max_energy_0_cycles + sqrt(pow(max_energy_0_cycles, 2) - 4 * (energy_decay_slope / 2) * (-
(energy_counting_p1_mWh + energy_counting_p2_mWh + energy_counting_p3_mWh) / 3) / 2)) / (2 * energy_decay_slope /
2);

  // Compute the maximum energy that the batteries can hold based on the equivalency in cycles. This is expressed in
  // mWh
  float max_energy_due_aging = energy_decay_slope * equivalency_cycles + max_energy_0_cycles;
  // Correct the measurement of the energy stored inside the batteries
  if (milli_volts_1 >= 16500) {  // Adjust when reached known maximum voltage. Get a known value.
    energy_pack1_corrected_mWs = ((long)max_energy_due_aging) * (60 * 60);
    energy_pack2_corrected_mWs = ((long)max_energy_due_aging) * (60 * 60);
    energy_pack3_corrected_mWs = ((long)max_energy_due_aging) * (60 * 60);
  } // if
  // Keep integrating the current over time
  energy_pack1_corrected_mWs = energy_pack1_corrected_mWs + milli_watts_3 * DeltaT / (1000);
  long energy_pack1_corrected_mWh = energy_pack1_corrected_mWs / (60 * 60);
  energy_pack2_corrected_mWs = energy_pack2_corrected_mWs + milli_watts_4 * DeltaT / (1000);
  long energy_pack2_corrected_mWh = energy_pack2_corrected_mWs / (60 * 60);
  energy_pack3_corrected_mWs = energy_pack3_corrected_mWs + milli_watts_5 * DeltaT / (1000);
  long energy_pack3_corrected_mWh = energy_pack3_corrected_mWs / (60 * 60);



  // Visualize the State of Charge using the LEDs integrated in the Shield
  // Evaluate if the batteries are fully charged, and turn on and off the green LED every two seconds
  // if the batteries are almost discharged, turn on and off the red LED every two seconds
  if (milli_volts_1 >= 16500) {
    if (Counter < 2) {
      // red LED off
      digitalWrite(redLEDpin, LOW);
      // green LED off
      digitalWrite(greenLEDpin, LOW);

      Counter += 1;
    } // if
    else {
      // red LED off
      digitalWrite(redLEDpin, LOW);
      // green LED on
      digitalWrite(greenLEDpin, HIGH);

      Counter = 0;

    } // else
  } // if
  else if (milli_volts_1 <= 13000) {
    if (Counter < 2) {
      // red LED off
      digitalWrite(redLEDpin, LOW);
      // green LED off
      digitalWrite(greenLEDpin, LOW);

      Counter += 1;
    }// if
    else {
      // red LED on
      digitalWrite(redLEDpin, HIGH);
      // green LED off
      digitalWrite(greenLEDpin, LOW);

      Counter = 0;

    } // else
  }// else if
  else {
    // red LED off
    digitalWrite(redLEDpin, LOW);
    // green LED off
    digitalWrite(greenLEDpin, LOW);
  } // else


  // Put together the string to be saved and displayed
  dataString += String(millivolts_skyassistant);
  dataString += "\t";
  dataString += String(rate_of_climb_m_s);
  dataString += "\t";
  dataString += String(MillSec);
  dataString += "\t";
  dataString += String(DeltaT);
  dataString += "\t";
  dataString += String(milli_volts_1);
  dataString += "\t";
  dataString += String(milli_amps_1);
  dataString += "\t";
  dataString += String(milli_watts_1);
  dataString += "\t";
  dataString += String(milli_watt_hour_1);
  dataString += "\t";
  dataString += String(milli_amps_2);
  dataString += "\t";
  dataString += String(milli_watts_2);
  dataString += "\t";
  dataString += String(milli_watt_hour_2);
  dataString += "\t";
  dataString += String(milli_amps_3);
  dataString += "\t";
  dataString += String(milli_watts_3);
  dataString += "\t";
  dataString += String(milli_watt_hour_3);
  dataString += "\t";
  dataString += String(energy_pack1_corrected_mWh);
  dataString += "\t";
  dataString += String(milli_amps_4);
  dataString += "\t";
  dataString += String(milli_watts_4);
  dataString += "\t";
  dataString += String(milli_watt_hour_4);
  dataString += "\t";
  dataString += String(energy_pack2_corrected_mWh);
  dataString += "\t";
  dataString += String(milli_amps_5);
  dataString += "\t";
  dataString += String(milli_watts_5);
  dataString += "\t";
  dataString += String(milli_watt_hour_5);
  dataString += "\t";
  dataString += String(energy_pack3_corrected_mWh);
  dataString += "\t";
  dataString += String(milli_volts_2);
  dataString += "\t";
  dataString += String(milli_amps_6);
  dataString += "\t";
  dataString += String(milli_watts_6);
  dataString += "\t";
  dataString += String(milli_watt_hour_6);
  dataString += "\t";
  dataString += String(max_energy_due_aging);
  dataString += "\t";
  dataString += String(energy_counting_p1_mWh);
  dataString += "\t";
  dataString += String(energy_counting_p2_mWh);
  dataString += "\t";
  dataString += String(energy_counting_p3_mWh);


  // Write the measurements in the SD card and display them.
  // Open file in the SD card
  dataFile = SD.open("datalog.txt", FILE_WRITE);
  if (dataFile) {
    // Write in the SD card
    dataFile.println(dataString);
    // Close the file in the SD card
    dataFile.close();
    // print to the serial port too:
    Serial.println(dataString);
  } // if
```

```
// if the file isn't open, pop up an error:
else {
  Serial.println("error opening datalog.txt");
} // else


// Put together the string to be sent over UDP
dataString = "";
dataString += "Reading_skyassistantRate_mV=";
dataString += String(millivolts_skyassistant);
dataString += "\r\n";
dataString += "Rate_of_climb_m_per_s=";
dataString += String(rate_of_climb_m_s);
dataString += "\r\n";
dataString += "Time_since_start_ms=";
dataString += String(MillSec);
dataString += "\r\n";
dataString += "Time_increment_ms=";
dataString += String(DeltaT);
dataString += "\r\n";
dataString += "Voltage_solar_panel_mV=";
dataString += String(milli_volts_1);
dataString += "\r\n";
dataString += "Current_solar_panel_mA=";
dataString += String(milli_amps_2);
dataString += "\r\n";
dataString += "Power_solar_panel_mW=";
dataString += String(milli_watts_2);
dataString += "\r\n";
dataString += "Energy_solar_panel_mWh=";
dataString += String(milli_watt_hour_2);
dataString += "\r\n";
dataString += "Current_pack1_mA=";
dataString += String(milli_amps_3);
dataString += "\r\n";
dataString += "Power_pack1_mW=";
dataString += String(milli_watts_3);
dataString += "\r\n";
dataString += "Energy_pack1_mWh=";
dataString += String(milli_watt_hour_3);
dataString += "\r\n";
dataString += "Energy_pack1_corrected_mWh=";
dataString += String(energy_pack1_corrected_mWh);
dataString += "\r\n";
dataString += "Current_pack2_mA=";
dataString += String(milli_amps_4);
dataString += "\r\n";
dataString += "Power_pack2_mW=";
dataString += String(milli_watts_4);
dataString += "\r\n";
dataString += "Energy_pack2_mWh=";
dataString += String(milli_watt_hour_4);
dataString += "\r\n";
dataString += "Energy_pack2_corrected_mWh=";
dataString += String(energy_pack2_corrected_mWh);
dataString += "\r\n";
dataString += "Current_pack3_mA=";
dataString += String(milli_amps_5);
dataString += "\r\n";
dataString += "Power_pack3_mW=";
dataString += String(milli_watts_5);
dataString += "\r\n";
dataString += "Energy_solar_pack3_mWh=";
dataString += String(milli_watt_hour_5);
dataString += "\r\n";
dataString += "Energy_pack3_corrected_mWh=";
dataString += String(energy_pack3_corrected_mWh);
dataString += "\r\n";
```

```
  dataString += "Voltage_load_mV=";
  dataString += String(milli_volts_2);
  dataString += "\r\n";
  dataString += "Current_load_mA=";
  dataString += String(milli_amps_6);
  dataString += "\r\n";
  dataString += "Power_load_mW=";
  dataString += String(milli_watts_6);
  dataString += "\r\n";
  dataString += "Energy_load_mWh=";
  dataString += String(milli_watt_hour_6);
  dataString += "\r\n";
  dataString += "Maximum_energy_due_aging_mWh=";
  dataString += String(max_energy_due_aging);
  dataString += "\r\n";
  dataString += "Energy_counting_pack1_mWh=";
  dataString += String(energy_counting_p1_mWh);
  dataString += "\r\n";
  dataString += "Energy_counting_pack2_mWh=";
  dataString += String(energy_counting_p2_mWh);
  dataString += "\r\n";
  dataString += "Energy_counting_pack3_mWh=";
  dataString += String(energy_counting_p3_mWh);
  dataString += "\r\n";

  char  SendBuffer[Buffer_size];
  dataString.toCharArray(SendBuffer, Buffer_size);

  // Send stream over UDP
  //Udp.beginPacket(Udp.remoteIP(), Udp.remotePort());
  Udp.beginPacket(remote_ip, remote_Port);
  Udp.write(SendBuffer);
  Udp.endPacket();

  // Frequency for sensing = 1 Hz
  delay(1000);

} // void loop()



// Write the header of the experimental data in the SD card and display it
void HeaderFile() {

  // Get the correct time from the RTC
  DateTime now = RTC.now();

  // Open the file to write in it
  dataFile = SD.open("datalog.txt", FILE_WRITE);

  // IF dataFile is open/created, write not only the time in it but also the variables and units to be measured
  if (dataFile) {
    String headerString = "";
    headerString +=  "This file puts the data recorded on datalog.txt in context.";
    // Write in the SD card
    dataFile.println(headerString);
    // print to the serial port too:
    Serial.println(headerString);
    headerString =  "datetime at the begining of the recording = ";
    headerString +=  String(now.month());
    headerString +=  "/";
    headerString +=  String(now.day());
    headerString +=  "/";
    headerString +=  String(now.year());
    headerString +=  " ";
    headerString +=  String(now.hour());
    headerString +=  ":";
    headerString +=  String(now.minute());
    headerString +=  ":";
    headerString +=  String(now.second());
    // Write in the SD card
    dataFile.println(headerString);
    // print to the serial port too:
    Serial.println(headerString);
    headerString =  "Data is organized as follows:";
    // Write in the SD card
    dataFile.println(headerString);
    headerString =  "(Voltage from skyassistant (mV) / Rate of climb (m/s)/ Time since the start (millisec) /  Time increment
(millisec) / milliVolts_1 / milliAmps_1 / milliWatts_1 /milliWattsHour_1/ milliAmps_2 / milliWatts_2 /milliWattsHour_2 /
milliAmps_3 / milliWatts_3 /milliWattsHour_3 / energy_pack1_corrected_mWh / milliAmps_4 / milliWatts_4 /milliWattsHour_4
/ energy_pack2_corrected_mWh /milliAmps_5 / milliWatts_5 /milliWattsHour_5 / energy_pack3_corrected_mWh /
milliVolts_2 / milliAmps_6 / milliWatts_6 /milliWattsHour_6/ max_energy_due_aging / energy_counting_mWh_p1 /
energy_counting_mWh_p2 / energy_counting_mWh_p3)";
    // Write in the SD card
    dataFile.println(headerString);
    // print to the serial port too:
    Serial.println(headerString);


    // Close the datalog.txt file in the SD card
    dataFile.close();

  } // if
  // if the file isn't open, pop up an error:
  else {
    Serial.println("error opening header.txt");
  } // else

} // void HeaderFile()
```

APPENDIX D. MATLAB CODES AND SIMULINK DIAGRAMS

Function `KF_h_V_estimates.m` *To Determine the Estimates of h and V and their Derivatives*

```
function Estimates = KF_h_V_estimates(dt,stdev_process,stdev_meas,...
    meas,u_t,Est_prev)
% This function applies a Linear Kalman Filter to estimate the values of
% the altitude and speed as well as their time derivatives

%% Define Update Equations

% State transition matrix
% State prediction
A = [1 dt dt^2/2 0 0 0; 0 1 dt 0 0 0; 0 0 1 0 0 0; 0 0 0 1 dt dt^2/2;...
    0 0 0 0 1 dt; 0 0 0 0 0 1];

% Input control matrix
% Expected effect of the input on the state
B = [0; 0; 0; 0; 0; 0];



% Measurement matrix
% The expected measurement
C = [1 0 0 0 0 0; 0 0 0 1 0 0];



%% Define previous estimates and the constant covariance matrices

% Previous estimates for the states and the covariance
x_t = [Est_prev(1); Est_prev(2); Est_prev(3); Est_prev(4); Est_prev(5);
Est_prev(6)];
E_t = [Est_prev(7) Est_prev(13) Est_prev(19) Est_prev(25) Est_prev(31)
Est_prev(37);
       Est_prev(8) Est_prev(14) Est_prev(20) Est_prev(26) Est_prev(32)
Est_prev(38);
       Est_prev(9) Est_prev(15) Est_prev(21) Est_prev(27) Est_prev(33)
Est_prev(39);
       Est_prev(10) Est_prev(16) Est_prev(22) Est_prev(28) Est_prev(34)
Est_prev(40);
       Est_prev(11) Est_prev(17) Est_prev(23) Est_prev(29) Est_prev(35)
Est_prev(41);
       Est_prev(12) Est_prev(18) Est_prev(24) Est_prev(30) Est_prev(36)
Est_prev(42)];



stdev_alt = stdev_process(1);
stdev_altd = stdev_process(2);
stdev_altdd = stdev_process(3);
stdev_vel = stdev_process(4);
stdev_veld = stdev_process(5);
```

```
stdev_veldd = stdev_process(6);

stdev_alt_meas = stdev_meas(1);
stdev_vel_meas = stdev_meas(2);

alt = meas(1);
vel = meas(2);


% Covariance matrices for noise
% Process covariance
E_x = [stdev_alt^2 0 0 0 0 0; 0 stdev_altd^2 0 0 0 0;...
    0 0 stdev_altdd^2 0 0 0; 0 0 0 stdev_vel^2 0 0;...
    0 0 0 0 stdev_veld^2 0; 0 0 0 0 0 stdev_veldd^2];

% Measurement covariance
E_z = [stdev_alt_meas^2 0; 0 stdev_vel_meas^2];

%% Equations of Kalman Filter

% Predict the next state
x_t = A * x_t + B * u_t;

% Predict the next covariance
E_t = A * E_t * A' + E_x;

% Kalman gain
K_t = E_t * C' * inv(C * E_t * C' + E_z);

% Update state estimate
x_t = x_t + K_t *([alt; vel] - C*x_t);

% Update covariance estimate
E_t = (eye(6) - K_t * C)*E_t;

%% Define output

% Make a vector out of the covariance matrix
E_t_reshape = reshape(E_t,1,36);

% Combine the estimates of the states with the covariance matrix in a
% single vector
Estimates = [x_t' E_t_reshape];

end
```

M-script t[JRL1]hat Extracts a 3D Elevation Map from the Google Elevation API

```
% This script builds the 3D map of the terrain getting data from the
Google
% elevation API.
% This script uses the xml2struct.m function.
%%%%%%% WARNING %%%%%%%%%%%%%
% Do not exceed the limits for requesting data from Google elevation API
% Otherwise Google will block the access

% The Elevation API has the following limits in place:

%     2,500 requests per 24 hour period.
%     512 locations per request.
%     25,000 total locations per 24 hour period.



%% Get information from user and declare variables

close all
clear all
clc

% Use format long to increase accuracy
format long

% Establish the limits of the area desired to be built. These numbers
must
% be in decimals

lat_min = 35.97973020660298;
long_min = -112.27502013090998;
lat_max = 36.24045347975275;
long_max = -111.93169737700373;


% Compute the distance
dist_lon = vdist(lat_min,long_min,lat_min,long_max)
dist_lat = vdist(lat_min,long_min,lat_max,long_min)



% Maximum number of samples in the area taking into account the maximum
% limit of 25,000 points per day and
% The number will be taken as a reference to create a square matrix
max_points = 25000;

% Determine number of samples to apply along latitude and along longitude
samples = floor(sqrt(max_points))

% Compute separation (resolution) of the samples in meters
res_lat = (dist_lat)/(samples - 1)  % meters per one sample
res_lon = (dist_lon)/(samples - 1) % meters per one sample
```

```
% Get a name for the map specified by the user
name = inputdlg('Type a name for your map');


% Initialize the latitude, longitude, and altitude variables to save the
% data
latitude = [];
longitude = [];
altitude = [];

% Create strings for the initial, final longitudes, and the number of
% samples
long_min_str = num2str(long_min,'%3.10f');
long_max_str = num2str(long_max,'%3.10f');
samples_str = num2str(samples,'%u');

% The output of Google elevation API is the elevation of a point in
meters,
% when given the latitude and longitude. The search can be done
specifying
% single points or paths specifying initial and final points together
with
% the number of samples that will be taken including the two endpoints.

% The area will be discretized in stripes using the searching method in
% paths. Along each path, the latitude will be constant and the longitude
% will change; and among paths, the latitude will change.

% Determine the increment in latitude to generate the paths
delta_lat = abs(lat_max-lat_min)/(samples-1);

%% Extract from Google Elevation API
for j=1:samples

    % Create the string of the new latitude for the path to retrieve the
elevation data
    new_lat = num2str(lat_min + delta_lat*(j-1),'%3.10f');

    % This is the format for requesting paths in the Google Elevation API
    %
http://maps.googleapis.com/maps/api/elevation/xml?path=46.52664349,15.041
694|46.52664349,15.199506&samples=158&sensor=true

    % Create an Document Object Model (DOM) from the xml file given by
the url of
    % Google Elevation API
    xDoc =
xmlread(['http://maps.googleapis.com/maps/api/elevation/xml?path=',new_la
t,',',long_min_str,'|',new_lat,',',long_max_str,'&samples=',samples_str,'
&sensor=true']);
```

```
    % This pause is just in case that the retrieving time from the Google
    % server is too long
    % pause(1)

    % This command is just a visual aid to know that the xml file is
being
    % retrieved from the Google server
    xmlwrite(xDoc)

    % This function extracts the data from the DOM file and create an
    % structure in MATLAB
    Elev_data = xml2struct(xDoc);


    % Get the information of latitude, longitude and elevation from every
    % sample point
    for i=1:samples
        %the number after comma in the struct result is the one that will
change from 1 to the number of samples
        latitude((j-1)*samples+i) =
str2num(Elev_data.ElevationResponse.result{1,i}.location.lat.Text);
        longitude((j-1)*samples+i) =
str2num(Elev_data.ElevationResponse.result{1,i}.location.lng.Text);
        altitude((j-1)*samples+i) =
str2num(Elev_data.ElevationResponse.result{1,i}.elevation.Text);
    end
end



% Reshape the data in matrices to be used with the surf command
X = reshape(longitude,samples,j);
Y = reshape(latitude,samples,j);
Z = reshape(altitude,samples,j);

% Plot using the surf command only when there is more than one path of
% points generated. This is because the surf command doesn't work when Z
is
% just a vector instead of a matrix.
if(size(Z,2) > 1)
    fig1 = figure('Color',[1 1 1]);
    surf(X,Y,Z,'FaceAlpha',1,'EdgeColor','none')%,'FaceColor',[1  0]);
    view(-69,62)
end

% Save the data to create further the maps without using information from
% the Google server
save(['Map_',name{1},'_','lat_min=',num2str(lat_min,'%3.7f'),...
    '_lat_max=',num2str(lat_max,'%3.7f'),'_long_min=',long_min_str,...
    '_long_max=',long_min_str,'_samples=',samples_str,'.mat'])
```

Script that Obtains the Model of a Thermal with Evolving Characteristics over Time

```
% This script plots the model of the thermal described by BENOIT
% CUSHMAN-ROISIN, which considers the evolution of the characteristics
% of the thermal over time

close all
clear all
clc

% Total bouyancy B = alpha*gT'V = g'V
B = 150;

% Time to be evalueted
t = 0:0.1:400;

% Radius of the thermal
R = 0.60*B^(1/4).*t.^(1/2);

% Altitude reached by the thermal
z = 2.41*B^(1/4).*t.^(1/2);

% Make a vector to get the points of theta around a circle
theta = 0:0.01:2*pi;

% Create the figure with the size of thermal and altitude at every step
in
% time
figure('Color',[1 1 1])
for i=1:length(z)
    x = R(i).*cos(theta);   % Create the points in x for every timestep
    y = R(i).*sin(theta);   % Create the points in y for every timestep
    subplot(2,2,[1 3])

plot3(x,y,z(i)*ones(1,length(x)),'Color',[mod(i,length(z))/length(z)...
        ,mod(i,length(z))/length(z)*0.5,mod(i,length(z))/length(z)*0.2])
    hold on
    subplot(2,2,2)

plot3(x,y,z(i)*ones(1,length(x)),'Color',[mod(i,length(z))/length(z)...
        ,mod(i,length(z))/length(z)*0.5,mod(i,length(z))/length(z)*0.2])
    hold on
    subplot(2,2,4)

plot3(x,y,z(i)*ones(1,length(x)),'Color',[mod(i,length(z))/length(z)...
        ,mod(i,length(z))/length(z)*0.5,mod(i,length(z))/length(z)*0.2])
    hold on
end

% Set up labels and orientation of the plots
subplot(2,2,[1 3])
grid on
```

```
xlabel('X, meters')
ylabel('Y, meters')
zlabel('Altitude, meters')
view(-39,18)
subplot(2,2,2)
grid on
xlabel('X, meters')
ylabel('Y, meters')
zlabel('Altitude, meters')
view(0,0)
subplot(2,2,4)
grid on
xlabel('X, meters')
ylabel('Y, meters')
zlabel('Altitude, meters')
view(0,90)
```

Script To Initialize the Probability Map

```
% This script calls the .mat file with the data obtained from the Google
% Elevation API. The slope and its derivative is determined to obtain the
% initial probability for the probability map. Additionally the 3D
% elevation map is used to plot the position of the glider during the
% simulation



%% Declaration of global variables
global prob_dist h_prob_map h_prob_map_fig long_middle_cell long_min...
    long_max lat_middle_cell lat_min lat_max thermal_id wp_index
visited...
    h_slovenia_fig wp_opt h_slovenia_g1 h_slovenia_g2 h_slovenia_g3
prob_max

%% Get the data from the mat file and generate the map of Slovenia
load('Map_Slovenia_20k_lat_min=46.3844530_lat_max=46.5644470_long_min=14.
9886810000_long_max=14.9886810000_samples=151.mat');


%% Get the information about the number of the gliders for the search
%  and how they will be distributed

% Get data from user
variables = {'Number of gliders for initial search (3 max)',...
    'Id for this glider (1, 2 or 3)'};
default = {'1','1'};
box_title = 'Data required for initial search';
g_info = inputdlg(variables, box_title, [1 20;1 20],default);

% Extract data from user input
ng = str2num(g_info{1}); % Number of gliders for the initial search
g_id = str2num(g_info{2}); % Number of this glider among the three

%% Get the slope of the terrain and plot it
```

```
m = samples;
n = j;

delta_lat_meters = res_lat;
delta_lon_meters = res_lon;

Z_diff_lon = diff(Z,1,1);
Z_diff_lat = diff(Z,1,2);

%Get the first derivative = slope
slope = sqrt((Z_diff_lon(:,1:end-1)/delta_lon_meters).^2 +...
    (Z_diff_lat(1:end-1,:)/delta_lat_meters).^2);
slope_angle = atand(slope);

% Compute the second derivative
Z_d_diff_lon = diff(Z,2,1);
Z_d_diff_lat = diff(Z,2,2);
d_slope = sqrt((Z_d_diff_lon(:,1:end-2)/delta_lon_meters).^2 +...
    (Z_d_diff_lat(1:end-2,:)/delta_lat_meters).^2);
d_slope_angle = atand(d_slope);

%% Form cells of approx. 300 meters and find their average slope in that
cell
% Find the number of samples to get cells of approximately 300 meters
s_to_300_lat = floor(300/res_lat);
s_to_300_lon = floor(300/res_lon);

for j=1:size(slope_angle,2)/s_to_300_lat-1

    for i=1:size(slope_angle,1)/s_to_300_lon-1
        slope_angle_middle_cell(i,j) = sum(sum(slope_angle(1+(i-1)*...
            s_to_300_lon:(s_to_300_lon + 1)+(i-1)*s_to_300_lon,1+...
            (j-1)*s_to_300_lat:(s_to_300_lat + 1) +(j-
1)*s_to_300_lat)))...
            /((s_to_300_lat + 1)*(s_to_300_lon + 1));
        d_slope_angle_middle_cell(i,j) = d_slope_angle((i-1)*...
            s_to_300_lon+s_to_300_lon,(j-1)*s_to_300_lat + s_to_300_lon);
        lat_middle_cell(i,j) = (Y(1,(j-1)*s_to_300_lat + ...
            (s_to_300_lat+1))-Y(1,(j-1)*s_to_300_lat + 1))/2 + ...
            Y(1,(j-1)*s_to_300_lat + 1);
        long_middle_cell(i,j) = (X((i-1)*s_to_300_lon +
(s_to_300_lon+1)...
            ,1)-X((i-1)*s_to_300_lon + 1,1))/2 + X((i-1)*...
            s_to_300_lon + 1,1);
    end

end


% Apply criteria to determine where the glider can gain altitude based on
% this: thermal soaring is generated in flat lands and at the hills

% Test different thresholds for the slope and its derivative
% prob = (slope_angle_middle_cell < 2 | slope_angle_middle_cell > 25);
% prob = (slope_angle_middle_cell < 2 | slope_angle_middle_cell > 30 |...
% d_slope_angle_middle_cell > 35);
% prob = (slope_angle_middle_cell > 25 | d_slope_angle_middle_cell > 35);
prob = (slope_angle_middle_cell < 3 | slope_angle_middle_cell > 30 );

% Generate the waypoints based on this:
index_wp = find(prob == 1);
wp_lat = lat_middle_cell(index_wp(index_wp >= numel(prob)*(g_id-1)/ng...
    & index_wp <= numel(prob)*(g_id)/ng));
wp_long = long_middle_cell(index_wp(index_wp >= numel(prob)*(g_id-
1)/ng...
    & index_wp <= numel(prob)*(g_id)/ng));

% Run TSP to get the optimal path
opt_index = TSP_index(lat_min,lat_max,long_min,long_max,wp_long,wp_lat);
[~,p_min] = min(sqrt((wp_long(opt_index)-15.1177).^2 + ...
    (wp_lat(opt_index)-46.471067).^2));
wp_pseudo_opt = [wp_lat(opt_index),wp_long(opt_index)];
wp_opt = [wp_pseudo_opt(p_min:end,:);wp_pseudo_opt(1:p_min-1,:)];



% Put an identifier to the cells. 0 for the cells with no prior
% probability, 1 for the cells with prior probability, and later 2 will
be
% used for the cells with thermals.
thermal_id = zeros(size(long_middle_cell));
% Assign 1 to the cells with prior
thermal_id(index_wp) = 1;

% Create a matrix to specify if the cell was visited or not
visited = zeros(size(long_middle_cell));

% Inititlize way point indexing to be used in the search stage
wp_index = 1;


%% Create the map of slovenia and the 3 airplanes
h_slovenia_fig = figure('Color',[1 1 1]);
surf(X,Y,Z,'FaceAlpha',0.8,'EdgeColor','none');
xlim([long_min long_max])
ylim([lat_min lat_max])
hold on

% Plot initial position for the three gliders at the runway
lat_g = 46.471067;
long_g = 15.1177;
alt_g = 500;

h_slovenia_g1 = plot3(long_g,lat_g,alt_g,'pr','MarkerSize',12,...
    'MarkerFaceColor','r');
h_slovenia_g2 = plot3(long_g,lat_g,alt_g,'pk','MarkerSize',12,...
    'MarkerFaceColor','k');
h_slovenia_g3 = plot3(long_g,lat_g,alt_g,'pm','MarkerSize',12,...
    'MarkerFaceColor','m');
xlabel('Longitude, deg','FontSize',12)
ylabel('Latitude, deg','FontSize',12)
zlabel('Elevation, m','FontSize',12)
view(-69,62)

%% Generate probability Map

% Convert the values of the probability map from logical to double and
% assign the initial probability to the cells keeping two separated
fields,
% one for the places with good priori probability and other for the
places
% with no good priori probability

prob_max = 0.9;
prob_dist = (prob + 0)*(prob_max)/sum(sum(prob));
ind_no_prior = find(prob == 0);
prob_dist(ind_no_prior) = (1 - prob_max)/length(ind_no_prior);

% Create figure for the probability map and plot it
h_prob_map_fig = figure('Color',[1 1 1]);

% Plot the bars and create the handle to color them later
h_prob_map = bar3(prob_dist);

% Color the map corresponding to the values of probability
for k = 1:length(h_prob_map)
zdata = get(h_prob_map(k),'Zdata');
set(h_prob_map(k),'Cdata',zdata,'EdgeColor','k','FaceAlpha',0.8);
end

% Manipulate axes properties
set(gca,'YTick',[0:length(long_middle_cell)/3:length(long_middle_cell)],'
YTickLabel',[long_min:(long_max-long_min)/3:long_max])
set(gca,'XTick',[0:length(lat_middle_cell)/3:length(lat_middle_cell)],'XT
ickLabel',[lat_min:(lat_max-lat_min)/3:lat_max])
xlabel('Latitude, deg','FontSize',12)
ylabel('Longitude, deg','FontSize',12)
zlabel('Probability','FontSize',12)
set(gca,'FontSize',12)
zlim([0 1])
view(195,40)
```

Function To Update the Probability Map

```
function Prob_map_3_gliders_norm(input)
% This function updates the probability map using a maximum of 3
% gliders in simulation

% Define the global variables
```

```
global prob_dist h_prob_map thermal_id visited prob_max

% Parameters of the sensor
alpha = 0.05; %False alarm P(sensor says target present|target absent)
beta = 0.01;  %Missed detection P(sensor says target absent|target
present)

% Combine signals from the three gliders
detection = [input(2) input(4) input(6)]; %Detection given by the
                                          % thermal state of the three
                                          % gliders (0, 1 or 2)
index = [input(1) input(3) input(5)]; % index of the cells where the
                                      % gliders are flying


% Compute the maximum probability that can hold a cell with a thermal
if(sum(sum(thermal_id == 2)) == 0)
    pmc = prob_max;
else
    pmc = prob_max/(sum(sum(thermal_id == 2)));
end

% This for loop goes through the index and detection of every glider
for i=1:length(index)

% Use the measurements of index and detection only if the corresponding
% glider is in the network determined by being in an index ~= 0
    if(index(i) ~= 0)
        % Mark the cell as visited
        visited(index(i)) = 1;

        % Compute the posteriori probability
        if (detection(i) ~=0)
            % Probability of target present given target detected
            P_tp = ((1-beta)*prob_dist(index(i)))/(alpha*(1-...
                prob_dist(index(i)))+(1-beta)*prob_dist(index(i)));
        elseif (detection(i) ==0)
            % Probability of target present given target not detected
            P_tp = (beta*prob_dist(index(i)))/((1-alpha)*(1-...
                prob_dist(index(i)))+beta*prob_dist(index(i)));
        end


        % Determine if a thermal is detected looking at increase of the
        % probability in the cell going above some threshhold
        if(P_tp > pmc && detection(i) ~= 0 && thermal_id(index(i))~= 2)

            % Mark the cell as having a thermal
            thermal_id(index(i)) = 2;
            % Determine the probability at the cells with thermals
            pmc = prob_max/(sum(sum(thermal_id == 2)));

            % Get the indices of the cells that have thermals on them
            % Assign the probability at the cells with thermals
            ind_2 = find(thermal_id == 2);
            prob_dist(ind_2) = pmc;

            % Get the indices of the cells that have no thermals on them
            % and assign the corresponding probability
            ind_0 = find(thermal_id == 0);% & visited == 0);
            prob_dist(ind_0) = (1-prob_max)*(1-prob_max)/(length(ind_0));
            ind_1 = find(thermal_id == 1);% & visited == 0);
            prob_dist(ind_1) = (1-prob_max)*(prob_max)/(length(ind_1));

        % Determine if the probability is bellow of some threshold to
        % determine the presence of a thermal; this portion is done at
        % every moment while the probability in a cell does not
        % increase because of the presence of a thermal
        elseif ( P_tp < pmc)

            % Find the indices of the cells corresponding to the thermal
            % ids and assign the probability to all the cells
            ind_2 = find(thermal_id == 2);
            ind_0 = find(thermal_id == 0);
            ind_0(ind_0 == index(i)) = [];
            ind_1 = find(thermal_id == 1);
            ind_1(ind_1 == index(i)) = [];

            if(numel(ind_2) == 0)
                prob_dist(ind_0) = prob_dist(ind_0) + (1-prob_max)*...
                    ((prob_dist(index(i)) - P_tp)/(length(ind_0)));
                prob_dist(ind_1) = prob_dist(ind_1) + (prob_max)*...
                    ((prob_dist(index(i)) - P_tp)/(length(ind_1)));
            else
                prob_dist(ind_2) = pmc +
prob_max*((prob_dist(index(i))...
                    - P_tp)/(length(ind_2)));
                prob_dist(ind_0) = prob_dist(ind_0) + (1-prob_max)*...
                    (1-prob_max)*((prob_dist(index(i)) - P_tp)/...
                    (length(ind_0)));
                prob_dist(ind_1) = prob_dist(ind_1) + (1-prob_max)*...
                    (prob_max)*((prob_dist(index(i)) - P_tp)/...
                    (length(ind_1)));
            end
            prob_dist(index(i)) = P_tp;
        end




        % Go through the values of the 3D plot bar to assign the values
of the
        % probability distribution matrix

        for o = 1:size(prob_dist,2)
            zdata = get(h_prob_map(o),'Zdata');
            k = 1;
            for j = 0:6:(6*size(prob_dist,1)-6)
                zdata(j+2:j+3,2:3) = prob_dist(k,o);
                k = k+1;
            end
            set(h_prob_map(o),'Zdata',zdata);
        end

        % Color the map corresponding to the values of probability
        for k = 1:length(h_prob_map)
            zdata = get(h_prob_map(k),'Zdata');

set(h_prob_map(k),'Cdata',zdata,'EdgeColor','k','FaceAlpha',0.8);
        end
    end
end


end
```

THIS PAGE INTENTIONALLY LEFT BLANK

SUPPLEMENTAL: PAPER ACCEPTED FOR PUBLICATION

A paper written by the author of this thesis and its advisors has been accepted for publication in the proceedings of the *19th World Congress of the International Federation of Automatic Control* (IFAC). Titled “Cooperative Autonomy of Multiple Solar-Powered Thermaling Gliders,” it gives a general overview of the capabilities developed for TaLEUAS. It is available as a supplemental to this thesis at the Dudley Knox Library of the Naval Postgraduate School.

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] F. L. Galvao, “A note on low draft bodies,” in *XI OSTIV Congress*, Lesyno, Poland, 1968, pp. 45–48.

[2] S. Dodbele and C. P. Van Dam, “Design of fuselage shapes for natural laminar flow,” NASA STI, Langley Research Center, Hampton, VA, TR-3970, 1968.

[3] L. M. Boermans and A. Van Garrel, “Design and wind tunnel test results of a flapped laminar flow airfoil for high-performance sailplane applications,” in ICAS convention, Anaheim, CA, 1994, pp. 1241–1247.

[4] M. S. Selig, J. J. Guglielmo, A. P. Broerer and P. Giguere, *Summary of Low-speed Airfoil Data*. Virginia Beach, VA: Soar Tech. Publications, 1995.

[5] L. Boermans and F. Nicolosi, “Sailplane fuselage and wing-fuselage junction design,” in *XXV Ostiv Congress*, S. Auban, France, 1997, section 1.3, pp. 1–9.

[6] M. S. Selig and J. J. Guglielmo, “High-lift low Reynolds number airfoil design,” *J. Aircraft,* vol. 34, pp. 72–79, 1997.

[7] A. Gopalarathnam and M. S. Selig, “Low-speed natural-laminar-flow airfoils: Case study in inverse airfoil design,” *J. Aircraft,* vol. 38, pp. 57–63, 2001.

[8] M. J. Allen, “Autonomous soaring for improved endurance of a small uninhabited air vehicle,” in *43rd AIAA Aerospace Sciences Meeting and Exhibit*, Reno, Nevada, 2005, pp. 1025–1037.

[9] M. J. Allen and V. Lin, “Guidance and control of an autonomous soaring UAV,” NASA, Edwards, CA, NASA/TM-2007-214611/REV1, Apr. 2007.

[10] D. J. Edwards, “Implementation details and flight test results of an autonomous soaring controller,” in *AIAA Guidance, Navigation and Control Conference and Exhibit*, Honolulu, Hawaii, 2008–7244.

[11] N. Akhtar, “Control system development for autonomous soaring,” Ph.D. dissertation, Dept. Aero. Eng., Cranfield Univ., Cranfield, Bedford, UK, 2010.

[12] K. Andersson, I. Kaminer, V. Dobrokhodov and V. Cichella, “Thermal centering control for autonomous soaring: Stability analysis and flight test results,” *J. Guidance, Control, and Dynamics,* vol. 35, no. 3, pp. 963–975, 2012.

[13] H. Reichmann, *Cross Country Soaring*, Hobbs, NM: Soaring Society of America, 1993.

[14] C. E. Hanson, “Cooperative autonomous thermal soaring for small uninhabited aerial vehicles,” M.S. thesis, Dept. Mech. Eng., California State Univ., Fresno, CA, 2008.

[15] K. Andersson, I. Kaminer, K. D. Jones, V. Dobrokhodov and D.J. Lee, “Cooperating uavs using thermal lift to extend endurance,” in *AIAA Infotech Aerospace Conference*, Seattle, WA, 2009–2043.

[16] C. G. O. Antal and S. Levi, “Adaptive autonomous soaring of multiple uavs using simultaneous perturbation stochastic approximation,” in *49th IEEE Conference on Decision and Control*, Atlanta, GA, 2010, pp. 3656–3661.

[17] J. A. Cobano, D. Alejo, S. Vera, G. Heredia and A. Ollero, “Multiple gliding uav coordination for static soaring in real time applications,” in *2013 IEEE International Conference on Robotics and Automation (ICRA)*, Karlsruhe, Germany, 2013, pp. 790–795.

[18] M. W. Hazard, “Unscented kalman filter for thermal parameter identification,” in *48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, Orlando, FL, 2010–179.

[19] S. R. Anton, “Energy harvesting for unmanned aerial vehicles,” unpublished.

[20] P. Barnes, R. Fishman and J. Tervamaki, “Electric uav using regenerative soaring and solar power (project proposal),” unpublished.

[21] A. Patil, V. Patil, D. W. Shin, J.W. Choi, D.S. Paik and S.J. Yoon, “Issue and challenges facing rechargeable thin film lithium batteries,” *Materials Research Bulletin*, vol. 43, pp. 1913–1942, 2008.

[22] “History of the battery.” Wikimedia Foundation, November 18, 2013. [Online]. Available: http://en.wikipedia.org/wiki/History_of_the_battery

[23] C.-X. Zu and H. Li, “Thermodynamic analysis on energy densities of batteries,” *Energy Environ. Sci.,* vol. 8, pp. 2614–2624, 2011.

[24] H. Lund, R. Nilsen, O. Salomatova, D. Skåre and E. Riisem, “The history highlight of solar cells (photovoltaic cells),” 2008. [Online]. Accessed 20 November 2013. Available: http://org.ntnu.no/solarcells/pages/history.php

[25] “Energy and Utilities,” Experience, Inc, 2013. [Online]. Accessed 18 November 2013. Available: https://www.experience.com/alumnus /article?channel_id=energy_utilities&source_page=additional_articles&article_id=article_1130427780670

[26] “NREL,” National Renewable Energy Laboratory. [Online]. Accessed 15 November. Available: http://www.nrel.gov/

[27] D. Doughty and E. P. Roth, “A general discussion of li ion battery safety,” *The Electrochemical Society’s Interface*, vol. 21, no. 2, pp. 37–44, 2012.

[28] M. Sudoh and J. Newman, “Mathematical model of the sodium/ironchloride battery,” *J. Electrochem. Soc.,* vol. 137, no. 3, pp. 876–883, 1990.

[29] R. Ramabadran and B. Mathur, “Effect of shading on series and parallel connected solar pv modules,” *Modern Applied Science,* vol. 3, no. 10, pp. 32–41, 2009.

[30] I. Büro, “Skymelody,” Skymelody. [Online]. Accessed 20 April 2014. Available: http://www.tek-variometer.de/englisch /Handb_Page_1/handb_page_1.htm

[31] K. J. Astrom and B. Wittenmark, *Adaptive Control*, 2nd ed. Addison-Wesley Publishing Company, 1995.

[32] D. Edwards, “Performance testing of RNR’s SB-XC using a Piccolo autopilot,.” March 14, 2008. [Online]. Accessed May 07, 2014. Available: http://www.xcsoaring.com/techpicts/edwards%20performance%20test.pdf

[33] “How to thermal better,” Sailing & Gliding. March, 2001. [Online]. Available: http://www.gliding.co.uk/bgainfo/aim%20higher/Howtothermalbetter.pdf

[34] *Glider Flying Handbook*, U. S. Department of Transportation, Federal Aviation Administration, Oklahoma, 2013.

[35] “Thermal,” Wikimedia Foundation, 26 March 2014. [Online]. Accessed 04 April 2014. Available: http://en.wikipedia.org/wiki/Thermal

[36] “Birds, Thermals, & Soaring Flight,” Aerospaceweb.org, 2012. [Online]. Accessed 20 February 2014. Available: http://www.aerospaceweb.org/question/nature/q0253.shtml

[37] M. J. Allen, “Updraft model for development of autonomous soaring uninhabited air vehicles,” NASA Dryden Flight Research Center.

[38] M. L. Zhenhua, “Modeling and simulation of autonomous thermal soaring with horizon simulation framework,” M.S. thesis, Dept. Aero. Eng., The Faculty of California Polytechnic State Univ., San Luis Obispo, CA., 2010.

[39] B. Cushman-Roisin, “Plumes and thermals,” in *Environmental Fluid Mechanics*, Hanover, NH, John Wiley & Sons, 2014, pp. 147–154.

[40] “Google Elevation API,” Google 11 March 2014. [Online]. Available: https://developers.google.com/maps/documentation/elevation/

[41] S. McIntyre, “More on Asphalt,” Climate Audit, 5 August 2007. [Online]. Available: http://climateaudit.org/2007/08/05/more-on-asphalt/

[42] “Code of Federal Regulations, General Operating and Flight Rules,” Federal Aviation Administration, 27 July 2004. [Online]. Accessed 2 June 2014. Available: http://rgl.faa.gov/Regulatory_and_Guidance_Library /rgFAR.nsf/0/934f0a02e17e7de086256eeb005192fc!OpenDocument

[43] G. Knier, “How do Photovoltaics work?,” NASA, 6 April 2011. [Online]. Accessed 20 May 2014. Available: http://science.nasa.gov/science-news/science-at-nasa/2002/solarcells/

[44] “How do solar panels work?,” Redarc electronics, 2014. [Online]. Accessed 20 May 2014. Available: http://www.redarc.com.au/solar/about/solarpanels/

[45] *C60 Solar Cell Mono Crystalline Silicon,* SunPower Corporation, San Jose, CA, 2010.

[46] “Arduino Mega 2560,” Arduino. [Online]. Accessed 21 May 2014. Available: http://arduino.cc/en/Main/ArduinoBoardMega2560

[47] “Specific Energy,” Wikimedia Foundation, 9 June 2014. [Online]. Available: http://en.wikipedia.org/wiki/Specific_energy

[48] “Lipo Batteries - The Good, the Bad and the Ugly,” Google sites. [Online]. Accessed 22 May 2014. Available: https://sites.google.com/site /mkokto2/training/lipo-batteries

[49] “Battery Space,” A A Portable Power Corp., 2013. [Online]. Accessed 2 June 2014. Available: http://www.batteryspace.com/PCM-with-Equilibrium-and-Temperature-protection-function-for-14.8V-Li-Ion.aspx

[50] “RnR Products,” RnR Products. [Online]. Accessed 2 June 2014. Available: http://www.rnrproducts.com/

[51] “Piccolo Autopilots,” UTC Aerospace Systems. [Online]. Accessed 2 June 2014. Available: http://www.cloudcaptech.com/piccolo_system.shtm

[52] V. Dobrokhodov, K. Jones and I. Kaminer, “Rapid flight and prototyping-steps toward cooperative mission-oriented capabilities,” in *American Control Conference*, Washington, DC, 2013, pp. 680–685.

[53] “MATLAB Coder,” The MathWorks, 2014. [Online]. Accessed 2 June 2014. Available: http://www.mathworks.com/products/matlab-coder/

[54] “Field Experimentation, Naval Postgraduate School,” Naval Postgraduste School, 2014. [Online]. Accessed 2 June 2014. Available: http://my.nps.edu/web/fx

[55] “Google Earth,” Google. [Online]. Accessed 15 May, 2014. Available: http://www.google.com/earth/

[56] “Solar cell,” Wikimedia Foundation, 15 May 2014. [Online]. Available: http://en.wikipedia.org/wiki/Solar_cell

[57] “Boeing 787 Dreamliner battery problems,” Wikimedia Foundation, 16 May 2014. [Online]. Available: http://en.wikipedia.org/wiki/Boeing_787_Dreamliner_battery_problems

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California